Strong Post-Compromise Secure Proxy Re-Encryption

Alex Davidson[†], Amit Deo[†], Ela Lee[†,1], Keith Martin

Royal Holloway, University of London

Abstract Proxy Re-Encryption (PRE), introduced by Bellare et. al in [6], allows a ciphertext encrypted using a key pk_i to be re-encrypted by a third party so that it is an encryption of the same message under a new key pk_j, without revealing the message. Post-Compromise Security (PCS) was first introduced for messaging protocols, and ensures that a ciphertext remains confidential even when past keys have been corrupted. We define PCS in the context of PRE, which ensures that an adversary cannot distinguish which ciphertext a re-encryption was created from even given the old secret key, potential old ciphertexts and update token used to perform the re-encryption. We argue that this formal notion accurately captures the most intuitive form of PCS. We give separating examples demonstrating how our definition is stronger than existing ones, before showing that PCS can be met using a combination of existing security definitions from the literature. In doing so, we show that there are existing PRE schemes that satisfy PCS. We also show that natural modifications of more practical PRE schemes can be shown to be PCS without relying on this combination of existing security definitions. Finally, we discuss the relationship between PCS with selective versus adaptive key corruptions, giving a theorem that shows how adaptive security can be met for certain re-encryption graphs.

1 Introduction

Cloud storage has become increasingly popular in recent years. Not long ago, cloud storage was mainly viewed as a useful backup for any files stored locally. However, in some situations cloud storage has started to become the for all storage and not just as a backup. One of the most prolific examples of this is in the increased popularity of media streaming platforms such as Netflix and Spotify, where clients subscribe to a service that gives them access to files on demand as opposed to storing such files locally. Other advantages come in the surge of smart devices, which no longer need much in-built storage as long as they have an internet connection.

[†] These authors are supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/K035584/1).

[1] Contact author: Ela.Lee.2010@live.rhul.ac.uk

Since the cloud is a third party, clients must encrypt their files to control ownership and access to their data. The difficulty comes when the key which a file is encrypted under needs changing. Key changes can happen for a variety of reasons. For instance: as a matter of good practice; or to enforce access control. For the former, NIST recommends regular key rotation [3], as does the Payment Card Industry Data Security Standard [29] and the Open Web Application Security Project (OWASP) [28]. For the latter, an organisation can choose which of its employees can access specific files by having those files encrypted under those keys. Should access need to be granted or revoked from someone, the files should be re-encrypted to a new key accordingly.

One trivial solution is to have the client download, decrypt, encrypt using the new key, and then re-upload the file. However, this can be very expensive in terms of both computational resources and storage, particularly for modern applications involving large databases, or if the client has limited processing capability. The primitive of Proxy Re-Encryption (PRE), introduced by Blaze et al. [6], presents a more elegant solution. In a PRE scheme, client creates an *update token* $\Delta_{i,j}$ using the current secret key sk_i and new public key pk_j and sends it to the server. The server can then use this token to re-encrypt the ciphertext, transforming it into an encryption of the same message which can now be decrypted using sk_j. The most basic security notion for PRE states that the server performing the re-encryption learns nothing about the underlying message.

Existing security definitions. The most basic security definition for PRE is that the server should not learn the underlying message or either of the keys – essentially an extension of Indistinguishable against Chosen Plaintext Attacks (IND-CPA) security factoring in re-encryption capabilities. Much existing work focused on creating definitions for CPA and Chosen Ciphertext Attacks (CCA) security for PRE variants, considering unidirectional/bidirectional PRE and single/multi-hop PRE. Such work rarely considers revocation as an application.

There is little work on requiring re-encryption to also re-randomise. [22] has one such definition IND-UPD for *updatable encryption schemes*, which are similar to symmetric PRE schemes but assume keys are updated sequentially. [16] also has similar definition UP RE-ENC. However, these definitions are limited to the key rotation application, and consider some schemes to be secure which we believe do not believe meet the intuition of PCS. We give further detail in Section 2.

Post-Compromise Security (PCS). The notion of PCS was first used in [13] for messaging protocols, which is informally defined as follows:

Definition 1 ([13]). *A protocol between Alice and Bob provides* Post-Compromise Security (PCS) *if Alice has a security guarantee about communication with Bob, even if Bob's secrets have already been compromised.*

As this definition gives the bare intuition, the 'security guarantee' is left open so that it can be tailored to the application, as we do in this work. This differs from *forward security*, which conveys that the compromise of future states does not affect the security of past ones. PCS conveys a scheme's ability to regain security after a session or party has been compromised, which has clear applications

involving revocation and key rotation (key life-cycles), where having access to the old key should not affect the security of a re-encrypted ciphertext.

Motivation for PCS PRE. One particular application of post-compromise PRE of interest is to complement PCS of messages in transit, by giving PCS security to backed up messages stored in the cloud. The Signal Protocol for encrypting messages in transit between two devices provides strong, modern, provable security guarantees including PCS [13]. However, most users expect to keep their messages when they lose their device or buy a new one; thus, popular Signal implementations such as WhatsApp back up client messages to public cloud services.Unfortunately, this backup is encrypted using a static encryption key. This means that while messages in transit have PCS, these properties are lost once messages are backed up. If an adversary compromises a device and obtains the static cloud backup key, they can retain this to compromise future messages once they are backed up.

Assuming that all message history is stored locally, the updated message history could be encrypted under a new key and re-uploaded at regular time intervals, but this will have a huge cost both in terms of computation and bandwidth, particularly as much messaging is done via smart-phones. A PRE scheme with PCS could be used instead, so that the PCS of messages in transit is extended to message backups.

1.1 Contributions

In this paper we set out the first formalisation of PCS for PRE schemes. Considering PCS for PRE, a strong definition would consider what happens when both old keys and update tokens used to perform a re-encryption are compromised. However, to date there is no security definition for that does this. We create a new definition which implies both unidirectionality, and that additional randomness beyond that given in the update token is added upon re-encryption. In our model, the adversary cannot distinguish a re-encrypted ciphertext given the old key, old ciphertexts and the token used to perform the re-encryption. In other words, we view a compromise as the loss of all previous public and secret states associated with a given ciphertext. Overall, we limit the information that must remain secret to the current secret key alone. Since we do not make as many assumptions on which algorithms are deterministic or on the flow of re-encryption operations, our definition applies to more general PRE schemes than similar definitions in the literature (see Section 2).

We analyse our model, proving several results that associate PCS with existing security models for PRE, as well as providing separating examples that distinguish PCS as a separate security characteristic in its own right. One of our major contributions is to show that a PRE scheme thatis both *source-hiding* and secure against *honest re-encryption attacks* also has PCS, meaning that several PRE schemes given by Fuchsbauer et al. [17] immediately satisfy PCS.

Unfortunately, these schemes require inefficient parameter choices, and are only proven secure for restricted re-encryption graphs (the structure by which

ciphertexts are moved between keys, see **??**) with a single source node, since the work of [17] focuses on achieving different security goals. We therefore give a new construction of a PRE scheme, pcBV-PRE, adapted from BV-PRE – the performant ring-Learning With Errors (RLWE) scheme of Polyakov et al. [30]. We prove that our adaptation achieves PCS and IND-CPA-security independently of the transformation that we define above, meaning that source-hiding is not necessary to achieve PCS. The advantage of this is that we can prove security for more general re-encryption graphs. Moreover, the lattice-based source-hiding schemes of [17] are forced into sub-optimal parameter choices that render their assumptions much stronger (with respect to the approximation factors of solving worst-case lattice problems) and much less efficient. Our new scheme is much more efficient since we leverage the speed of the original construction with minimal changes.

Finally, we show that achieving PCS with adaptive key compromises is possible via the transformation of [17] using adaptively IND-CPA-secure and source-hiding PRE schemes. Such schemes currently only exist for certain graphs (trees and chains) with polynomial security loss. We leave achieving adaptively-secure PCS for more general graphs as interesting future work.

Paper structure. We begin by reviewing related work and necessary preliminaries in Sections 2–4. Motivated by our discussion of related work, we define a new notion of PCS for PRE in Section 5 before demonstrating how our definition relates to those already in the literature using separating examples to demonstrate that our definition is stronger than previous work, then showing how a combination of existing definitions leads to PCS. In Section 6, we give an explicit, efficient construction by modifying the lattice-based BV-PRE scheme [30] to show that our notion of PCS can be satisfied by natural extensions of current practical PRE schemes. Finally, in Section 7 we discuss the relationship between selective and adaptive security for PCS using the work of [17].

2 Related work

There are many different types of PRE schemes in the literature that satisfy various properties. For example, PRE schemes can be bidirectional, meaning that a single update token for Alice and Bob allows to re-encrypt ciphertexts not only from Alice to Bob, but from Bob to Alice too. Alternatively, PRE schemes can be unidirectional in the sense that an update token only allows one to update ciphertexts in one direction. There is also a distinction between single-hop and multi-hop PRE schemes. Single-hop schemes only allow ciphertexts to be re-encrypted once whereas multi-hop schemes allow for multiple re-encryptions.

The basic security definition for PRE was first given in [6] for bidirectional schemes. It similar to IND-CPA security but adapted to allow for re-encryption capabilities, meaning that the proxy does not learn the underlying message during the re-encryption process. In this work we refer to such notions as PRE-CPA, to avoid confusion with IND-CPA (for Public Key Encryption (PKE) schemes) and PKE-CPA (for PRE schemes). Unidirectional PRE schemes were introduced

by Ateniese et al. [2] together with an equivalent security notion. A definition for Indistinguishable against Chosen Ciphertext Attacks (IND-CCA) security for PRE first appears in [9] for bidirectional single-hop (ciphertexts can only be re-encrypted once) schemes, and is the first definition where the adversary can adaptively corrupt secret keys. A definition for CCA security for unidirectional schemes is given in [23].

It was an open problem for many years to create a PRE scheme which is both unidirectional and multi-hop, with single-hop constructions emerging as the means of providing unidirectionality in the meantime. Multi-hop schemes are necessary for both key rotation and dynamic access control. Unidirectional schemes are necessary for applications where the trust relationship between Alice and Bob is not symmetrical, such as if Bob is more senior to Alice and therefore has access to more sensitive files. Whilst most existing literature considers directionality as a class of PRE schemes, it was been defined explicitly as a formal security definition in [21]. The first constructions for PRE-CPA secure unidirectional, multi-hop PRE schemes were given using program obfuscation [11, 19], until Gentry [18] gave a generic construction using Fully Homomorphic Encryption (FHE).

A significant recent development in recent times is the notion of Honest Re-encryption Attacks (HRA), introduced by Cohen [12] which allows the adversary to re-encrypt non-challenge ciphertexts from uncorrupted keys to corrupted ones, unlike in PRE-Indistinguishable against Chosen Plaintext Attacks (PRE-CPA) where no such re-encryptions are permitted. Cohen also defines *re-encryption simulatability* and demonstrates IND-CPA-secure schemes which have this property are Indistinguishable against Honest Re-encryption Attacks (IND-HRA)-secure.

Fuchsbauer et al. present relations between selective and adaptive security in [17]. They expand on Cohen's work, defining *weak key privacy* and *source-hiding* and show how these can reduce security against adaptive HRAs to security against CPA, with sub-exponential loss for some re-encryption graphs (see Section 3 for a definition of re-encryption graphs). In particular, they show quasi-polynomial loss in security when lifting to adaptive corruption for trees and chains, and exponential loss otherwise. However it is also worth noting that their proofs rely on re-encryption trees having only one source, which limits the number of applications the scheme is guaranteed to be secure in.

The original notion of post-compromise security is from messaging protocols [13], to reflect the idea that compromise of past sessions does not impact the confidentiality of future ones. PCS builds on the idea that re-encryption should fully re-randomise a ciphertext upon re-encryption. To see this, consider the *key encapsulation approach* to PRE: encryption involves generating a symmetric key k and performing a symmetric encryption $C_1 \xleftarrow{\$} \mathcal{SE}.\mathsf{Enc}(k, m)$, before encrypting k using a public-key PRE scheme $C_0 \leftarrow \mathcal{PRE}.\mathsf{Enc}(\mathsf{pk}, k)$. For re-encryption, the ciphertext header C_0 is re-encrypted so it is now an encryption of k under some specified pk', whilst C_1 remains unchanged. This method is popular as it allows for a hybrid approach, where encryption of the message is symmetric and the header uses public-key encryption. With this approach, key-scraping attacks are possible. A malicious user simply retains the symmetric encryption key k and

can derive the message regardless of how often the ciphertext is re-encrypted. It may be unrealistic for a malicious revoked user to download all the plaintexts due to storage constraints, as is the case for subscriber-based streaming platforms. However, as symmetric keys are typically much shorter than the plaintexts, it is much more realistic that a malicious subscriber could perform the above attack.

Whilst re-randomisation is not generally a goal of PRE schemes, it has been covered recently in related work [9, 16, 22]. [9] discusses a notion for re-randomising ciphertexts (which they call *unlinkability*) only in the context of creating a CCA definition which considers re-randomisation. Other notions of re-randomisation mainly exist in the related areas of *key rotation* and *updatable encryption*. These differ from proxy re-encryption in that they are symmetric schemes, that re-encryption happens sequentially from k_i to k_{i+1}, and they name full re-randomisation as a necessary security goal.

Definitions conveying this include IND-UPD security [22] and UP-REENC security [16]. IND-UPD aims to cover PCS for *updatable encryption* schemes. However, an updatable encryption scheme can be bidirectional and still be IND-UPD, as bidirectionality invokes the additional winning condition that the adversary cannot have learned any update tokens used to create a challenge ciphertext. We give an asymmetric version of IND-UPD in Appendix B, which we use in Section 5.4 to demonstrate that our notion is stronger. UP-REENC security does not give the adversary the token used to create the challenge re-encryption, and therefore does not cover full compromise of the user who generated the update token. Their construction, ReCrypt allows an adversary who has learned the update token to reverse the re-encryption of the challenge ciphertext. Other related work models PCS by giving a bound on the amount of information the adversary maintains of the old ciphertext [21, 27]. However, such schemes are weaker than the notion we define particularly if considering revoked users storing parts of the original ciphertext and colluding, or a user relinquishing ownership of a file. We create a stronger definition in which schemes must be unidirectional, and the adversary knows the update token used to create the challenge re-encryption.

3 Preliminaries

In this section, we give the preliminaries for proxy re-encryption, including some common security definitions and an explanation of directed re-encryption graphs. Whilst we stick to the asymmetric setting in the body of this work, we give symmetric variants for important definitions in Appendix A for easier comparison with related work in the symmetric setting.

Definition 2. *A* Proxy Re-Encryption (PRE) *scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda) \rightarrow$ *params: Outputs a set of public parameters, including the message space and ciphertext space. Note that params is input to every subsequent algorithm, but we leave it out for compactness of notation. We often omit the* Setup *algorithm for the same reason.*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: Generates a public-private key pair.*
- $\mathsf{Enc}(\mathsf{pk}, m) \to C$*: Encrypts a message* m *using a public key* pk*, producing a ciphertext* C*.*
- $\mathsf{Dec}(\mathsf{sk}, C) \to m' \cup \perp$*: Decrypts a ciphertext* C *to produce either an element of the message space* m' *or an error symbol* $\perp$*.*
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \to \Delta_{i,j} \cup \perp$*: Takes a secret key* sk_i *and public key* pk_j *and outputs an* update token $\Delta_{i,j}$*, or* $\perp$ *when* $i = j$*. This last condition is often left out of constructions for compactness.*
- $\mathsf{ReEnc}(\Delta_{i,j}, C) \to C'$*: Takes a ciphertext* C *under* pk_i *and outputs a new ciphertext* C' *under* pk_j*.*

A PRE scheme is correct *if, for all* $m \in \mathcal{M}, (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$*, then:*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) \to m$$

and if, for all $C \in \mathcal{C}$ *such that* $\mathsf{Dec}(\mathsf{sk}_i, C) \to m$*, then:*

$$\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ReEnc}(\Delta_{i,j}, C)) \to m$$

where $(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ *and* $\Delta_{i,j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$.

Note that some PRE constructions have a *correctness bound* – a limit on the number of re-encryptions that are possible before the resulting ciphertext fails to decrypt properly. We shall see this in Section 6.

3.1 Re-encryption graphs

We often use a directed re-encryption graph (DRG) when discussing the security of PRE schemes. A DRG tracks queries the adversary $\mathcal{A}$ makes during a security game to represent re-encryptions that $\mathcal{A}$ can make locally. The DRG consists of nodes v_i that represent key pairs, and directed edges $\overrightarrow{e}_{i,j}$ which are added when a $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$ query is made. Using update tokens, the adversary can locally re-encrypt challenge ciphertexts. Therefore, if a challenge ciphertext is an encryption under pk_i, and there exists a sequence of tokens going from i to j, then both sk_i and sk_j are considered challenge keys. Represented using a graph $\mathcal{DRG}$, if $\mathcal{DRG}$ contains a path from v_i to v_j and sk_i is a challenge key, then so is sk_j. The DRG is often used to enforce the *trivial win condition* which states that the adversary cannot query oracles to reveal a challenge under a corrupted key. This is a standard condition in all PRE security definitions. Figure 1 gives a pictorial representation of this.

For example, for simply rotating keys the resulting graph will be a chain, as is assumed in *updatable encryption* [7, 22] and *key rotation for authenticated*

Note that some definitions of a PRE scheme have an additional input ℓ to indicate a *level* the ciphertext should be at. In this work, we leave out ℓ unless discussing schemes and results that use levelling explicitly.

if a scheme is bidirectional, then edges added would be directionless. In this work we mainly focus on unidirectional schemes

Figure 1: An example directed re-encryption graph, $\mathcal{DRG}$. If a challenge is learned under pk_6 and $\mathcal{A}$ has learned tokens $\Delta_{6,5}$ and $\Delta_{6,7}$, then $\mathsf{pk}_5, \mathsf{pk}_6$ are also considered challenge keys. Therefore, token queries that lead to paths from challenge keys to corrupted ones such as $\Delta_{6,1}$ or $\Delta_{5,2}$ result in a trivial win. The graph on the right gives some examples of what tokens the adversary can and cannot learn next.

encryption [16], whereas some access control hierarchies may lead to trees. Some results in existing work [17] mainly apply to certain types of graph. Re-encryption graphs are often used in security proofs, subject to graphs being *acyclic*. We make this assumption throughout this paper.

3.2 Common oracles

Most definitions use the same oracles, which we define in Figure 2 for compactness. The main variations between definitions are how the challenge oracle $\mathsf{O}_{\mathsf{challenge}}$ is defined, and sometimes whether $\mathsf{O}_{\mathsf{ReEnc}}$ affects the DRG. We therefore define these in each individual game. Games often keep track of lists updated by the oracles, namely a list of challenge keys $\mathcal{K}_{\mathsf{chal}}$, corrupted keys $\mathcal{K}_{\mathsf{corrupted}}$, oracle-generated ciphertexts $\mathcal{C}_{\mathsf{honest}}$, challenge ciphertexts $\mathcal{C}_{\mathsf{chal}}$ and oracle-generated tokens $\mathcal{T}_{\mathsf{honest}}$.

$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda)$

$\kappa = \kappa + 1$

$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$

$\boxed{\mathcal{DRG}.\mathsf{add}\ (v_\kappa)}$

return pk_κ

$\mathsf{O}_{\mathsf{Corrupt}}(i)$

$\mathcal{K}_{\mathsf{corrupted}}.\mathsf{add}\ (\mathsf{sk}_i)$

return sk_i

$\mathsf{O}_{\mathsf{Enc}}(i, m)$

$C \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m)$

$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (i, C)$

$\boxed{\mathcal{C}_{\mathsf{msg}}[(i, C)] = m}$

return C

$\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$

if $\mathcal{DRG} \cup \vec{e}_{i,j}$ is cyclic:

  return $\perp$

$\Delta_{i,j} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$

$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\ (i, j, \Delta_{i,j})$

$\boxed{\mathcal{DRG}.\mathsf{add}\ (\vec{e}_{i,j})}$

return $\Delta_{i,j}$

Figure 2: Common oracles used in security games for PRE. κ is the number of keys in the game. $\boxed{\text{Boxed values}}$ indicate steps to update lists that may not be used depending on the game. The lists a particular game uses are indicated in the game's setup phase.

$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda)$ increments the number of keys κ, generates a new key pair and adds a new vertex to $\mathcal{DRG}$ to represent this key pair before returning the public key. $\mathsf{O}_{\mathsf{Corrupt}}(i)$ adds sk_i to the set of corrupted keys $\mathcal{K}_{\mathsf{corrupted}}$ and returns it

to the adversary. $\mathsf{O}_{\mathsf{Enc}}(i, m)$ encrypts the message under pk_i, appends the list of honestly-created ciphertexts $\mathcal{C}_{\mathsf{honest}}$ with (i, C) and returns the ciphertext. Sometimes we also need to consider a lookup table which can give the message used to create an honestly-generated ciphertext, to use proof techniques where challenge ciphertexts are replaced with fresh encryptions when the key is unknown. In this case, $\mathcal{C}_{\mathsf{msg}}$ is used. $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$ creates the update token, appends the list of honestly-generated update tokens $\mathcal{T}_{\mathsf{honest}}$ with $(i, j, \Delta_{i,j})$, adds the edge $\overrightarrow{e}_{i,j}$ to $\mathcal{DRG}$ and returns the update token.

In our syntax, the restrictions on what tokens the adversary is allowed to learn is not enforced by oracles (as in other work), but instead by the list of challenge keys $\mathcal{K}_{\mathsf{chal}}$ being updated using $\mathcal{DRG}$ at the end of the game. The following function is used to update the set of challenge keys:

$\mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$

$\forall i$ such that $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{chal}}$:
  $\forall j$ such that $\exists$ a path from v_i to v_j in $\mathcal{DRG}$:
    $\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ \mathsf{sk}_j$
return $\mathcal{K}_{\mathsf{chal}}$

We enforce the trivial win condition by calling $\mathsf{UpdateChallengeKeys}$ at the end of the game, and checking that no challenge keys have been corrupted ($\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupted}} = \emptyset$).

3.3 Indistinguishability definitions

Here we give a number of indistinguishability notions, covering both traditional PKE and message indistinguishability in PRE. For the present we only concern ourselves with security for selective key corruptions. Security with adaptive key corruptions discussed in Section 7.

IND-CPA in PKE and PRE. IND-CPA-security is a well-known notion in public-key encryption which states that given a ciphertext, an adversary cannot distinguish which of two messages it is an encryption of.

$\mathsf{PKE\text{-}CPA}_{\mathcal{A}}^{b,\mathcal{PKE}}(1^\lambda)$

$\kappa = 0$
$b' \leftarrow \mathcal{A}_1^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{challenge}}^{\mathsf{PKE\text{-}CPA}}}(1^\lambda)$
return b'

$\mathsf{O}_{\mathsf{challenge}}^{\mathsf{PKE\text{-}CPA}}(i, m_0, m_1)$

if $|m_0| \neq |m_1|$: **return** $\perp$
$C \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_b)$
return C

Figure 3: The PKE-CPA game. This is usually called the IND-CPA game, but we make the distinction here for indistinguishability of chosen-plaintext attacks for PKE schemes and PRE schemes.

Definition 3. *A PKE scheme $\mathcal{PKE}$ is* ϵ-Indistinguishable against Chosen Plaintext Attacks (ϵ-PKE-CPA-secure) *if for all Probabilistic Polynomial-Time (*PPT*) adversaries $\mathcal{A}$:*

$$\left|\Pr\left[\mathsf{PKE\text{-}CPA}_{\mathcal{A}}^{0,\mathcal{PKE}}(1^\lambda)=1\right]-\Pr\left[\mathsf{PKE\text{-}CPA}_{\mathcal{A}}^{1,\mathcal{PKE}}(1^\lambda)=1\right]\right|\leq\epsilon$$

where PKE-CPA *is defined in Figure 3. If ϵ is negligible as parameterised by the security parameter λ, then we say the scheme is* Indistinguishable against Chosen Plaintext Attacks (PKE-CPA-secure).
A PRE scheme $\mathcal{PRE}$ is ϵ-IND-CPA secure if the PKE scheme given by $\mathcal{PKE} = \{\mathcal{PRE}.\mathsf{KeyGen}, \mathcal{PRE}.\mathsf{Enc}, \mathcal{PRE}.\mathsf{Dec}\}$ is ϵ-PKE-CPA-secure.

Extending IND-CPA for PRE. The most common definition of security for PRE builds on the previous definition, stating that the scheme should still be IND-CPA secure when given the additional functionality of re-encryption. This reflects that the proxy should not learn the message during the re-encryption process. We now give a definition for CPA security for PRE schemes.

$\mathsf{PRE\text{-}CPA}_{\mathcal{A}}^{b,\mathcal{PRE}}(1^\lambda)$

$\mathcal{K}_{\mathsf{chal}}, \mathcal{K}_{\mathsf{corrupted}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{DRG} = \emptyset$
$\kappa = 0, \mathsf{called} = \mathsf{false}$
$state \leftarrow \mathcal{A}_0^{\mathsf{O}_{\mathsf{KeyGen}},\mathsf{O}_{\mathsf{Corrupt}},\mathsf{O}_{\mathsf{Enc}},\mathsf{O}_{\mathsf{ReEnc}}}(1^\lambda)$
$b' \leftarrow \mathcal{A}_1^{\mathsf{O}_{\mathsf{KeyGen}},\mathsf{O}_{\mathsf{ReKeyGen}},\mathsf{O}_{\mathsf{ReEnc}}^{\mathsf{PRE\text{-}CPA}},\mathsf{O}_{\mathsf{challenge}}^{\mathsf{PRE\text{-}CPA}}}(1^\lambda, state)$
$\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupted}} \neq \emptyset$: **return** 0
return b'

$\mathsf{O}_{\mathsf{ReEnc}}^{\mathsf{PRE\text{-}CPA}}(C, i, j, [\Delta_{i,j}])$

if $\Delta_{i,j}$ given **AND** $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$:
  return $\perp$
if $\Delta_{i,j}$ not given :
  $\Delta_{i,j} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\mathcal{DRG}.\mathsf{add}\ \vec{e}_{i,j}$
$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C)$
return C'

$\mathsf{O}_{\mathsf{challenge}}^{\mathsf{PRE\text{-}CPA}}(i, m_0, m_1)$

if $\mathsf{called} = \mathsf{true}$ **OR** $|m_0| \neq |m_1|$:
  return $\perp$
$C \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_b)$
$\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ \mathsf{sk}_i$
$\mathsf{called} \leftarrow \mathsf{true}$
return C

Figure 4: The PRE-CPA game – an extension of IND-CPA which accounts for re-encryption. $\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}$ are as defined in Figure 2.

Definition 4. *A PRE scheme $\mathcal{PRE}$ is said to be* (selectively) ϵ-PRE-Indistinguishable against Chosen Plaintext Attacks-secure (ϵ-PRE-CPA-secure) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left|\Pr\left[\mathsf{PRE\text{-}CPA}_{\mathcal{A}}^{0}(1^\lambda)=1\right]-\Pr\left[\mathsf{PRE\text{-}CPA}_{\mathcal{A}}^{1}(1^\lambda)=1\right]\right|\leq\epsilon,$$

where PRE-CPA *is defined in Figure 4.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme is (selectively) PRE-Indistinguishable against Chosen Plaintext Attacks-secure (PRE-CPA-secure).

Whilst the above definition is based on the one given in [17], our formulation is slightly different as we account for there being multiple possible tokens per key pair, meaning $\mathsf{O}_{\mathsf{ReEnc}}$ allows $\mathcal{A}$ to input an honestly-generated update token as opposed to only having indexes as input. Note that the $\mathcal{DRG}$ is created by adding an edge whenever $\mathsf{O}_{\mathsf{ReEnc}}$ is called.

Honest Re-encryption Attacks. Recently, a stronger notion than PRE-CPA security has been introduced which allows the adversary to re-encrypt non-challenge ciphertexts to any key, as long as the ciphertexts were honestly generated. Cohen formalised these as *Honest Re-encryption Attacks (HRA)* [12] but the same idea is also used elsewhere [22]. This motivates IND-HRA-security [12]. We base our formulation on IND-ENC-security [22], HRA-security [12] and IND-CPA-security [30].

$\mathsf{IND\text{-}HRA}^{b,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$

$\mathcal{K}_{\mathsf{chal}}, \mathcal{K}_{\mathsf{corrupted}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{DRG} = \emptyset$
$\kappa = 0, \mathsf{called} = \mathsf{false}$
$state \leftarrow \mathcal{A}_0^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReEnc}}}(1^\lambda)$
$b' \leftarrow \mathcal{A}_1^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{O}^{\mathsf{IND\text{-}HRA}}_{\mathsf{ReEnc}}, \mathsf{O}^{\mathsf{IND\text{-}HRA}}_{\mathsf{challenge}}}(1^\lambda, state)$
$\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupted}} \neq \emptyset$: **return** 0
return b'

$\mathsf{O}^{\mathsf{IND\text{-}HRA}}_{\mathsf{ReEnc}}(C, i, j, [\Delta_{i,j}])$

if $(i, C) \notin \mathcal{C}_{\mathsf{honest}}$:
  return $\perp$
if $\Delta_{i,j}$ given **AND** $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$:
  return $\perp$
if $\Delta_{i,j}$ not given :
  $\Delta_{i,j} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C')$
if $(i, C) \in \mathcal{C}_{\mathsf{chal}}$:
  $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C'), \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_j)$
return C'

$\mathsf{O}^{\mathsf{IND\text{-}HRA}}_{\mathsf{challenge}}(i, m_0, m_1)$

if $|m_0| \neq |m_1|$ **OR** $\mathsf{called} = \mathsf{true}$:
  return $\perp$
$C \xleftarrow{\$} \mathsf{ReEnc}(\mathsf{pk}_i, m_b)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (i, C)$
$\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (i, C)\ \ \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_i)$
$\mathsf{called} \leftarrow \mathsf{true}$
return C

Figure 5: The IND-HRA game. Like the HRA model [12], it allows re-encryptions of non-challenge ciphertexts to compromised keys using $\mathsf{O}_{\mathsf{ReEnc}}$.

Definition 5. *A PRE scheme $\mathcal{PRE}$ is said to be* (selectively) ϵ-Indistinguishable against Honest Re-encryption Attacks-secure (ϵ-IND-HRA-secure) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left|\Pr\left[\mathsf{IND\text{-}HRA}^0_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{IND\text{-}HRA}^1_{\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \epsilon,$$

where $\mathsf{IND\text{-}HRA}^{b,\mathcal{PRE}}_{\mathcal{A}}$ *is defined in Figure 5.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme is (selectively) Indistinguishable against Honest Re-encryption Attacks-secure (IND-HRA-secure).

We discuss security with respect to adaptive key corruptions in Section 7.

Theorem 1. *IND-HRA $\implies$ PRE-CPA $\implies$ PKE-CPA.*

As each game builds directly on the last but giving the adversary access to more information, the proof of this theorem follows trivially.

4 Existing Definitions that Re-randomise

Recall from the key encapsulation example that re-randomisation is necessary for PCS. The definitions covered in Section 3.3 are not designed to examine re-randomisation, and are therefore too weak for key revocation considerations. In this section, we discuss IND-UPD security for asymmetric encryption, as IND-UPD it is the closest definition for PCS for PRE. We also give a definition for source-hiding which we will show in Theorem 2 gives PCS when a scheme is also IND-HRA-secure. Whilst these definitions consider key corruption to an extent, they do not consider an adversary who can learn the old secret key and update token and therefore they do not cover full compromise of the user who generated the update token.

4.1 Asymmetric IND-UPD

IND-UPD [22, definition 3] is a post-compromise security notion for updatable encryption schemes, which are a variant of symmetric PRE schemes. We consider a version adapted to the public-key setting, for easier comparison to our definitions. We call this pkIND-UPD, for which we give the main points here, and a full description in Appendix B.

In the $\mathsf{pkIND\text{-}UPD}$ game, key updates happen sequentially. The challenge oracle outputs a re-encrypted ciphertext and the adversary must guess which ciphertext it is a re-encryption of. Challenge ciphertexts are updated whenever a new key is generated, but only given to the adversary if the oracle $\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{LearnChal}}$ is called. One of the winning conditions is given in $\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{ReEnc}}$ is that when ReEnc is deterministic, the adversary cannot have re-encrypted either of the potential challenge ciphertexts $\bar{C}_0, \bar{C}_1$. Another notable condition is that the adversary cannot learn the update token going towards the key that the challenge is given

under, which is enforced as a condition in $\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{LearnTok}}$. The final constraint is related to directionality, that if the scheme is bidirectional then $\mathcal{A}$ cannot have learned any tokens leading from corrupted keys to challenge keys. We will address these points when we come to build our own definition in Section 5.

4.2 Source-hiding

Fuchsbauer et al. [17] define *source-hiding* as a component for demonstrating that PRE security with selective key corruptions can imply security with adaptive key corruptions in restricted re-encryption circumstances. Informally, in a source-hiding scheme it is possible to create a fresh encryption of a message that is indistinguishable from a re-encrypted ciphertext that is an encryption of the same message. This means re-encrypted ciphertexts reveal no history as to the keys they were previously encrypted under, or similarities between components of previous ciphertexts.

We give a formal description of the game defining the source-hiding property in Figure 6. Our formulation generalises the original definition in [17] by allowing the adversary to receive κ keypairs rather than 1. Moreover, as before, we allow the adversary to query multiple re-key tokens between any key pairs of their choice.

$\mathsf{SH}^{b,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda, 1^\kappa, 1^L)$

$\{(\mathsf{pk}_\iota, \mathsf{sk}_\iota) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)\}_{\iota\in[\kappa]}$
$\mathcal{T}_{\mathsf{honest}} = \emptyset$
$b' \leftarrow \mathcal{A}^{\mathsf{O}^{\mathsf{SH}}_{\mathsf{challenge}}, \mathsf{O}_{\mathsf{ReKeyGen}}}(1^\lambda, \{(\mathsf{pk}_\iota, \mathsf{sk}_\iota)\}_{\iota\in[\kappa]})$
return b'

$\mathsf{O}^{\mathsf{SH}}_{\mathsf{challenge}}(i, j, \Delta^*_{i,j}, m^*, \ell^*)$

if $[(i, j, \Delta^*_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$ **OR** $\ell^* + 1 > L]$:
  return $\perp$
$C^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m^*, \ell^*)$
$C^{(0)} \xleftarrow{\$} \mathsf{ReEnc}(\Delta^*, C^*)$
$C^{(1)} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_j, m^*, \ell^* + 1)$
return $(C^*, C^{(b)})$

Figure 6: Experiments for the source-hiding property. Here, ℓ denotes a *level* for the ciphertext to be encrypted at – essentially the number of times C has been re-encrypted. This is important for noisy PRE schemes, but ignored for PRE schemes without levelling. L is the number of times a ciphertext can be re-encrypted without breaking the correctness conditions (the correctness bound).

Definition 6. *A PRE scheme $\mathcal{PRE}$ is said to be* ϵ-source-hiding (ϵ-SH) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left|\Pr\left[\mathsf{SH}^{0,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda, 1^\kappa, 1^L) = 1\right] - \Pr\left[\mathsf{SH}^{1,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda, 1^\kappa, 1^L) = 1\right]\right| \leq \epsilon$$

where $\mathsf{SH}^{b,\mathcal{PRE}}_{\mathcal{A}}$ is defined in Figure 6.

If ϵ is negligible as parameterised by the security parameter, then we say the scheme is source-hiding (SH).

5 Strong PCS for PRE

In this section, we create a definition for strong PCS for public-key PRE schemes. We begin by justifying our motivation, explaining why we believe existing definitions are insufficient for PCS.

5.1 Motivation for a new definition

We have two main motivations for creating a new definition. The first is that there is currently no definition that implies unidirectionality and ciphertext re-randomisation. We believe that the distinction of whether a PRE scheme is unidirectional or bidirectional is vital in the post-compromise scenario to model the corruption of used update tokens, and hence we define PCS to explicitly mean that schemes meeting the definition must be unidirectional. The second motivation is a matter of existing definitions inherently assuming which of the algorithms in the PRE scheme are probabilistic. We explain how this introduces problems when considering a post-compromise scenario. We now go into both of these motivations in more detail.

Explicit unidirectionality. We use the most relevant definition in existing literature to make our case. IND-UPD [22] places restrictions based on inferable information, as defined by the [22] notions of directionality:

- When sk_j cannot be derived from sk_i and $\Delta_{i,j}$ (LT-unidirectional)
- When sk_j can be derived from sk_i and $\Delta_{i,j}$ (LT-bidirectional).

In the LT-unidirectional case, the adversary can acquire re-encryption tokens from a corrupted key to a challenge key, but not the other way around. In the LT-bidirectional case, the adversary is additionally prevented, by definition, from learning tokens from challenge keys to corrupted keys or vice versa. This means that for bidirectional schemes, the adversary queries tokens in such a way that the resulting re-encryption graphs form disjoint sub-graphs – one containing corrupted keys and the other containing challenge keys. Proving security is therefore reduced to proving that unrelated, randomly-generated keys cannot be used to infer information about an encrypted message. We consider this too restrictive for the intuition of PCS.

Since the derivability of the new secret key is implicit information to the game, meeting the definition itself says nothing about directionality and therefore whether update token compromise is of concern. We believe that it is better to have directionality given explicitly as it allows practitioners to understand the security of a scheme more clearly.

Assuming probabilistic algorithms. There appear to be only two existing security definitions which explicitly consider re-randomisation [16, 22]. The [16]

The general understanding of unidirectionality is not so strong - the new key does not necessarily have to be derivable, but the token and old key should lead to the message being learned.

definition of a *key rotation scheme* assumes that ReKeyGen is randomised but ReEnc is deterministic. This leads to a necessary condition that the update token used to create the challenge re-encryption cannot be learned by the adversary, otherwise the adversary could use it to re-encrypt the input ciphertexts locally and compare this to the challenge to win the game. The adversary is allowed to learn other tokens going from corrupted to challenge keys using an oracle $\mathsf{O}_{\mathsf{ReKeyGen}}$, but not the exact token used to perform the re-encryption. This means UP-REENC [16] does not model compromise of the update token used. For this reason, it is important that new randomness is not just introduced via the update token to prevent trivial downgrading of the challenge ciphertext if the adversary compromises the update token used.

In the [22] definition of an *updatable encryption scheme*, the opposite assumption is made – that ReEnc is randomised and ReKeyGen is deterministic. This means that for keys $\mathsf{sk}_i, \mathsf{pk}_j$, there is only one update token $\Delta_{i,j}$. This is reflected in their IND-UPD security game (and pkIND-UPD) by having all tokens generated at the start of the game and later having oracles reveal tokens to the adversary. This is less fitting for schemes where ReKeyGen is randomised and there are multiple tokens per key pair. More importantly, such an assumption rules out the ways in which secret keys are masked in the update token, which is important for PCS. The BV-PRE scheme is an example of this, where knowledge of the key 'pk_j' together with $\Delta_{i,j}$ can be used to derive sk_i. Another example are ElGamal-based symmetric PRE schemes (e.g. [6, 22]) where update tokens have the form $\Delta_{i,j} = \mathsf{sk}_j/\mathsf{sk}_i$. Clearly, given the update token, compromise of the old key leads to compromise of the new key. Introducing some randomness gives a means of masking the new key. It also means that the client does not need to fully rely on the proxy to be assured of new randomness, which is more appropriate for some trust scenarios.

For constructions where randomness is added in both ReKeyGen and ReEnc, neither definition is suitable. It is therefore of interest to create a security notion for PCS which factors in the possibility that both the ReKeyGen and ReEnc algorithms are probabilistic.

5.2 Post-Compromise Security

In our notion of Post-Compromise Security (PCS), an adversary $\mathcal{A}$ chooses two ciphertexts and a re-encryption token, and receives a challenge ciphertext which is a re-encryption of one of the original ciphertexts created using the specified token. $\mathcal{A}$ attempts to distinguish which ciphertext was re-encrypted. This models the compromise of all key-related material prior to the challenge re-encryption.

Unlike some previous definitions, we allow $\mathcal{A}$ to corrupt the secret key with which the update token was generated. As in IND-HRA security, we also allow $\mathcal{A}$ to re-encrypt honestly(oracle)-generated non-challenge ciphertexts to corrupted keys. Challenges are obtained via the challenge oracle $\mathsf{O}^{\mathsf{PC}}_{\mathsf{challenge}}$ which will only

Interestingly, other works such as [17] take a similar approach to the adversary learning update tokens, despite their assuming randomised tokens.

accept as input honestly-generated ciphertexts of the same length, and honestly-generated update tokens. The challenger maintains lists to enforce this: $\mathcal{C}_{\mathsf{honest}}$ and $\mathcal{T}_{\mathsf{honest}}$ respectively.

Here we present a formal definition PCS for general PRE. In the first stage, the adversary $\mathcal{A}$ can access a key corruption oracle $\mathsf{O}_{\mathsf{Corrupt}}$ and in the second stage they can access the challenge oracle $\mathsf{O}^{\mathsf{PC}}_{\mathsf{challenge}}$ and re-encryption oracle $\mathsf{O}_{\mathsf{ReKeyGen}}$. As in previous definitions, $\mathcal{K}_{\mathsf{corrupted}}$ and $\mathcal{C}_{\mathsf{chal}}$ are also maintained, which record corrupted keys and challenge ciphertexts, respectively and used to enforce the trivial win condition at the end of the game.

$\mathsf{PostComp}^{b,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$

$\mathcal{K}_{\mathsf{chal}}, \mathcal{K}_{\mathsf{corrupted}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}}, \mathcal{C}_{\mathsf{msg}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{DRG} = \emptyset$
$\kappa = 0, \mathsf{called} = \mathsf{false}$
$state \leftarrow \mathcal{A}_0^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}^{\mathsf{PC}}_{\mathsf{ReEnc}}}(1^\lambda)$
$b' \leftarrow \mathcal{A}_1^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{O}^{\mathsf{PC}}_{\mathsf{ReEnc}}, \mathsf{O}^{\mathsf{PC}}_{\mathsf{challenge}}}(1^\lambda, state)$
$\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupted}} \neq \emptyset$: **return** 0
return b'

$\mathsf{O}^{\mathsf{PC}}_{\mathsf{ReEnc}}(C, i, j, [\Delta_{i,j}])$

if $\Delta_{i,j}$ given :
  if $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$
else : $\Delta_{i,j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
if $(i, C) \notin \mathcal{C}_{\mathsf{honest}}$: **return** $\perp$
$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C')$
$\mathcal{C}_{\mathsf{msg}}[(j, C')] = \mathcal{C}_{\mathsf{msg}}[(i, C)]$
if $(i, C) \in \mathcal{C}_{\mathsf{chal}}$:
  $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C'), \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_j)$
return C'

$\mathsf{O}^{\mathsf{PC}}_{\mathsf{challenge}}(C_0, C_1, i, j, \Delta_{i,j})$

if $|C_0| \neq |C_1|$ **OR** $\mathsf{called} = \mathsf{true}$: **return** $\perp$
if $(i, C_0), (i, C_1) \notin \mathcal{C}_{\mathsf{honest}}$ **OR** $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$
$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C_b)$
$\mathcal{C}_{\mathsf{msg}}[(j, C')] = \mathcal{C}_{\mathsf{msg}}[(i, C_b)]$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C'), \mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C'), \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_j)$
$\mathsf{called} \leftarrow \mathsf{true}$
return C'

Figure 7: The PostComp game. This reflects full compromise of the old secret key and update token used to perform the re-encryption.

Definition 7. *A PRE scheme $\mathcal{PRE}$ is said to have* (selective) ϵ-Post-Compromise Security (ϵ-PCS) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left|\Pr\left[\mathsf{PostComp}^0_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{PostComp}^1_{\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \epsilon,$$

where $\mathsf{PostComp}\mathcal{A}^{b,\mathcal{PRE}}$ *is defined in Figure 7.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme has (selective) Post-Compromise Security (PCS).

We give a definition of PCS for symmetric PRE schemes in Appendix A, and for adaptive key corruptions in Section 7.

5.3 Basic observations

In the following lemma, we show that it is necessary for PCS to have randomness incorporated into re-encryption.

Lemma 1. *No PRE scheme where* ReEnc *is deterministic has PCS.*

Proof. If ReEnc is deterministic then $\mathcal{A}$ can submit $(C_0, C_1, i, j, \Delta_{i,j})$ to $\mathsf{O}_{\mathsf{challenge}}$ to learn challenge C'. Then $\mathcal{A}$ can locally compute $C'_0 \leftarrow \mathsf{ReEnc}(\Delta_{i,j}, C_0)$ and compare this with C' – if they match then output $b' = 0$, otherwise output $b' = 1$.

Lemma 2. *PCS* $\implies$ *unidirectional.*

Proof. We show that if a scheme is bidirectional then it cannot have PCS. Bidirectionality implies that an update token $\Delta_{i,j}$, can be used to derive $\Delta_{j,i}$. The adversary $\mathcal{A}$ proceeds as follows: $\mathcal{A}$ first corrupts a key sk_i, then creates two ciphertexts $C_0 \leftarrow \mathsf{O}_{\mathsf{Enc}}(i, m_0), C_1 \leftarrow \mathsf{O}_{\mathsf{Enc}}(i, m_1)$ and update token $\Delta_{i,j} \leftarrow \mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$, before submitting $(C_0, C_1, i, j, \Delta_{i,j})$ to the challenge oracle. $\mathcal{A}$ then computes $\Delta_{j,i}$ from $\Delta_{i,j}$, and uses this to compute $C'' \leftarrow \mathsf{ReEnc}(\Delta_{j,i}, C')$ where C' is the received challenge re-encryption to obtain the challenge ciphertext under key pk_i, before decrypting using sk_i to win the game.

This means an easy way to disprove PCS for existing schemes is to show bidirectionality.

5.4 Separating examples

We now demonstrate the relationship between PCS and existing security notions and constructions by means of a number of separating examples.

Lemma 3. *Let* $\mathcal{PRE}$ *be a PRE scheme where* $\mathsf{ReKeyGen}$ *is deterministic. If* $\mathcal{PRE}$ *has PCS, then it is pkIND-UPD-secure.*

Proof. The $\mathsf{PostComp}$ adversary can easily simulate the $\mathsf{pkIND\text{-}UPD}$ game. In oracles, whenever there is a KeyGen, the simulator calls $\mathsf{O}_{\mathsf{KeyGen}}$, and whenever there is a $\mathsf{ReKeyGen}$, the simulator calls $\mathsf{O}_{\mathsf{ReKeyGen}}$, and otherwise follows the procedures as described with the exception of the challenge oracle. For the challenge oracle $\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{challenge}}$, the adversary replaces the challenge ciphertext with the output from $\mathsf{O}^{\mathsf{PC}}_{\mathsf{challenge}}(C_0, C_1, \kappa - 1, \kappa)$. This simulates the $\mathsf{pkIND\text{-}UPD}$ game perfectly, and therefore the adversary will be able to win $\mathsf{pkIND\text{-}UPD}$ with the same advantage as for $\mathsf{PostComp}$.

Lemma 4. *pkIND-UPD-security* $\not\implies$ *PCS.*

Proof. Let $\mathcal{PRE}$ be a PRE scheme where ReEnc is deterministic, and it is pkIND-UPD-secure. By Lemma 1, this scheme is not post-compromise secure.

Lemma 5. *IND-HRA-security $\not\Longrightarrow$ PCS.*

Proof. Let $\overline{\mathcal{PRE}} = (\overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}}, \overline{\mathsf{ReKeyGen}}, \overline{\mathsf{ReEnc}})$ be a IND-CPA-secure PRE scheme, and let $(\mathsf{symKeyGen}, \mathsf{symEnc}, \mathsf{symDec})$ be an IND-CPA symmetric encryption scheme. We now define a PRE scheme using the key encapsulation method as follows:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk}) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \overline{\mathsf{KeyGen}}(1^\lambda)$
- $\mathsf{Enc}(\mathsf{pk}, m) \to (c_0, c_1) : k \xleftarrow{\$} \mathsf{symKeyGen}(1^\lambda), c_0 \leftarrow \overline{\mathsf{Enc}}(\mathsf{pk}, k), c_1 \leftarrow \mathsf{symEnc}(k, m)$
- $\mathsf{Dec}(\mathsf{sk}, c) \to m' : k' \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0), m' \leftarrow \mathsf{symDec}(k', c_1)$
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \to \Delta_{i,j} : \Delta_{i,j} \leftarrow \overline{\mathsf{ReKeyGen}}(\mathsf{sk}_i, \mathsf{pk}_j)$
- $\mathsf{ReEnc}(\Delta_{i,j}, c) \to (c'_0, c'_1) : c'_0 \leftarrow \overline{\mathsf{ReEnc}}(\Delta_{i,j}, c_0), c'_1 = c_1$

This scheme is also IND-HRA, but is not PCS. An adversary $\mathcal{A}$ can submit two ciphertexts to the challenge oracle, and compare the second part of the challenge re-encryption with the submitted ciphertexts to win the game.

Lemma 6. *PCS $\not\Longrightarrow$ IND-HRA-security.*

Proof. Let $\overline{\mathcal{PRE}} = (\overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}}, \overline{\mathsf{ReKeyGen}}, \overline{\mathsf{ReEnc}})$ be a PRE scheme that is IND-HRA-secure and has PCS. We use it to construct the following PRE scheme:

- $\mathsf{KeyGen}(1^\lambda) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \overline{\mathsf{KeyGen}}(1^\lambda)$
- $\mathsf{Enc}(\mathsf{pk}, m) : C \leftarrow (m, \overline{\mathsf{Enc}}(\mathsf{pk}, m))$
- $\mathsf{Dec}(\mathsf{sk}, C) : m' \leftarrow \overline{\mathsf{Dec}}(\mathsf{sk}, C_1)$
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) : \Delta_{i,j} \leftarrow \overline{\mathsf{ReKeyGen}}(\mathsf{sk}_i, \mathsf{pk}_j)$
- $\mathsf{ReEnc}(\Delta_{i,j}, C) : C'_0 \leftarrow \overline{\mathsf{Enc}}(\mathsf{pk}_j, 0), C'_1 \leftarrow \overline{\mathsf{ReEnc}}(\Delta_{i,j}, C_1)$

Clearly this scheme is not IND-HRA secure, as fresh ciphertexts contain the plaintext. However the scheme has PCS, as re-encryptions C'_1 will be unrelated to C_1, since $\overline{\mathcal{PRE}}$ has PCS, and the creation of C'_0 is independent of both C_0 and $\Delta_{i,j}$.

Since PCS does not imply any security notion concerning confidentiality of the message, confidentiality definitions must be proven separately for in order to demonstrate that a PRE scheme is useful in practice.

5.5 PCS via source-hiding and IND-HRA

In this section we show that a PRE scheme that is both source-hiding and IND-HRA-secure also has PCS.

Theorem 2 (main). *Let $\mathcal{PRE}$ be a PRE scheme that satisfies ϵ_1-IND-HRA-security and is ϵ_2-SH. Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be PPT algorithms that are attempting to succeed in the $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}$, $\mathsf{SH}_{\mathcal{B}}^{\mathcal{PRE}}$ and $\mathsf{IND\text{-}HRA}_{\mathcal{C}}^{\mathcal{PRE}}$ security games, respectively. Let $\mathcal{A}$ have advantage ϵ in $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}$. Then:*

$$\epsilon \leq 2\epsilon_2 + \epsilon_1 < \mathsf{negl}(\lambda),$$

for a negligible function $\mathsf{negl}(\lambda)$, and thus $\mathcal{PRE}$ has PCS.

Proof. Let $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE},b}$ refer to the experiment in Figure 7, where the choice of b is made explicit. We prove this theorem using a sequence of hybrid steps, we start with the execution of $\mathcal{PRE}$ in $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE},0}$. We show that via a sequence of reductions that this situation is computationally indistinguishable from the case where $\mathcal{A}$ witnesses the execution in $b = 1$.

Firstly, we define a new oracle:

$\overline{\mathsf{O}_{\mathsf{challenge}}}(C_0, C_1, i, j, \Delta_{i,j})$

if $|C_0| \neq |C_1|$ **OR** $\mathsf{called} = \mathsf{true}$: **return** $\perp$
if $(i, C_0), (i, C_1) \notin \mathcal{C}_{\mathsf{honest}}$ **OR** $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$
$(m_0, m_1) \leftarrow (\mathcal{C}_{\mathsf{msg}}[(i, C_0)], \mathcal{C}_{\mathsf{msg}}[(i, C_1)])$
if $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$ **OR** $\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupted}}$: **return** $\perp$
$C' \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_j, m_b)$
$\mathcal{C}_{\mathsf{msg}}[(j, C')] = m_b$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C'), \mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C'), \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_j)$
$\mathsf{called} \leftarrow \mathsf{true}$
return C'

Let $\mathsf{O}_{\mathsf{challenge},b}$ and $\overline{\mathsf{O}_{\mathsf{challenge},b}}$ be the executions of the oracles where the choice of $b \in \{0, 1\}$ is made explicit.

- Game_0: This is the original $\mathcal{PRE}$ construction in $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE},0}$.
- Game_1: Replace outputs from the oracle $\mathsf{O}_{\mathsf{challenge},0}$ with outputs from the oracle $\overline{\mathsf{O}_{\mathsf{challenge},0}}$.
- Game_2: Replace $\overline{\mathsf{O}_{\mathsf{challenge},0}}$ with $\overline{\mathsf{O}_{\mathsf{challenge},1}}$.
- Game_3: Replace $\overline{\mathsf{O}_{\mathsf{challenge},1}}$ with $\mathsf{O}_{\mathsf{challenge},1}$.

It is not hard to see that Game_3 is identical to the execution in the case of $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE},1}$. Therefore, if we can bound the advantage in distinguishing the game transitions above by a negligible function, then the probability of distinguishing in $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE},b}$ is also negligible.

Lemma 7. *The advantage in distinguishing* Game_0 *and* Game_1 *is bounded by* ϵ_2.

Proof. This proof holds in the case where $\mathcal{PRE}$ has a maximum of $\kappa = \mathsf{poly}(\lambda)$ nodes. $\mathcal{B}$ receives key pairs $(\mathsf{pk}_\iota, \mathsf{sk}_\iota)_{\iota \in [\kappa]}$ and re-encryption tokens $\{\Delta_{i,j}\}_{(i,j) \in [\kappa]^2}$ for $\mathcal{PRE}$. Let $\overline{\mathsf{O}_{\mathsf{challenge}}}$ be the challenge oracle that $\mathcal{B}$ has access to in $\mathsf{SH}_{\mathcal{B}}^{\mathcal{PRE},c}$.

Then $\mathcal{B}$ instantiates the oracles $\mathsf{O}_{\mathsf{KeyGen}}$, $\mathsf{O}_{\mathsf{Corrupt}}$, $\mathsf{O}_{\mathsf{Enc}}$, $\mathsf{O}_{\mathsf{ReKeyGen}}$, $\mathsf{O}_{\mathsf{ReEnc}}$ in the same way as in $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE},b}$. Let $\mathcal{C}_{\mathsf{honest}}$ and $\mathcal{C}_{\mathsf{msg}}$ be the sets that $\mathcal{B}$ keeps track of the queries made to $\mathsf{O}_{\mathsf{Enc}}$ and $\mathsf{O}_{\mathsf{ReEnc}}$. For a query $(C_0^*, C_1^*, (i^*, j^*), \Delta_{i,j}^*)$ made to the oracle $\mathsf{O}_{\mathsf{challenge},0}$ by $\mathcal{A}$, then $\mathcal{B}$ does the following:

- **if** $|C_0^*| \neq |C_1^*|$: **return** $\perp$;
- **if** $(i^*, C_0^*), (i^*, C_1^*) \notin \mathcal{C}_{\mathsf{honest}}$: **return** $\perp$;
- **if** $(i^*, j^*, \Delta_{i^*,j^*}^*) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$;

– **if** $\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupted}}$: **return** $\perp$;
– $(m_0^*, m_1^*) \leftarrow \mathcal{C}_{\mathsf{msg}}$;
– $C_0^\dagger \xleftarrow{\$} \overline{\mathsf{O}_{\mathsf{challenge}}}(1^\lambda, (i^*, j^*), (m_0^*, C_0^*))$;
– $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C_0^\dagger)$; $\mathcal{C}_{\mathsf{msg}}[(j, C_0^\dagger)] = m_0^*$; $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C_0^\dagger)$;
– **return** $C_0^\dagger$.

In the case of $c = 0$, then $C_0^\dagger \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C_0^*)$. In the case of $c = 1$, then $\overline{\mathsf{O}_{\mathsf{challenge}}}$ returns $C_0^\dagger \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_{j^*}, m_0^*)$. Consequently, when $c = 0$ then $\mathcal{B}$ simulates $\mathsf{O}_{\mathsf{challenge},0}$; for $c = 1$ then $\mathcal{B}$ simulates $\overline{\mathsf{O}_{\mathsf{challenge},0}}$. Now, let $\Pr\left[0 \xleftarrow{\$} \mathcal{A}[\mathsf{Game}_d]\right] = \epsilon_d$ be the probability that $\mathcal{A}$ outputs $b' = 0$ for $d \in \{0, 1\}$. Then:

$$|\Pr\left[0 \xleftarrow{\$} \mathcal{A}[\mathsf{Game}_0]\right] - \Pr\left[0 \xleftarrow{\$} \mathcal{A}[\mathsf{Game}_1]\right]| = |\delta_0 - \delta_1| = \delta$$

represents the differences in the probabilities in the two games. Then, let ϵ_0 be the probability that $\mathcal{A}$ outputs 0 in the simulation of $\mathcal{B}$. If $\epsilon_0 \approx \delta_0$ then $\mathcal{B}$ outputs $c' = 0$ to $\mathsf{SH}_{\mathcal{B}}^{\mathcal{PRE},c}$. Otherwise, if $\epsilon_0 \approx \delta_1$ then $\mathcal{B}$ outputs $c' = 1$.

Notice that if δ is noticeably large, then $\mathcal{B}$ can win the $\mathsf{SH}_{\mathcal{B}}^{\mathcal{PRE},c}$ with the same advantage. In other words, $\mathcal{B}$ can run sequential subroutines using the adversary $\mathcal{A}$ and build up a probability distribution based on the answers of $\mathcal{A}$. If the answers correlate more to the probability inferred by δ_0 then $\mathcal{B}$ answers 0, and otherwise 1.

Therefore, $\delta < \mathbf{Adv}^{\mathsf{SH}}$; where $\mathbf{Adv}^{\mathsf{SH}}$ is the advantage of $\mathcal{B}$ in SH. By the assumption that $\mathcal{PRE}$ is ϵ_2-SH, then we have that $\delta < \epsilon_2$.

Lemma 8. *The advantage in distinguishing* Game_1 *and* Game_2 *is bounded by* ϵ_1.

Proof. Let $\overline{\mathsf{O}_{\mathsf{challenge}}}$ be the challenge oracle $\mathsf{O}_{\mathsf{challenge}}$ that $\mathcal{B}$ has access to in $\mathsf{IND\text{-}HRA}_{\mathcal{B}}^{\mathcal{PRE},b}(1^\lambda)$. Firstly, $\mathcal{B}$ simulates the oracles $\mathsf{O}_{\mathsf{KeyGen}}$, $\mathsf{O}_{\mathsf{Corrupt}}$, $\mathsf{O}_{\mathsf{Enc}}$, $\mathsf{O}_{\mathsf{ReKeyGen}}$, $\mathsf{O}_{\mathsf{ReEnc}}$ using the oracles in $\mathsf{IND\text{-}HRA}_{\mathcal{B}}^{\mathcal{PRE},c}(1^\lambda)$. Then, for a query $(C_0^*, C_1^*, (i^*, j^*), \Delta_{i,j}^*)$ made to $\overline{\mathsf{O}_{\mathsf{challenge},b}}$ by $\mathcal{A}$, $\mathcal{B}$ does the following:

– **if** $|C_0^*| \neq |C_1^*|$: **return** $\perp$;
– **if** $(i^*, C_0^*), (i^*, C_1^*) \notin \mathcal{C}_{\mathsf{honest}}$: **return** $\perp$;
– **if** $(i^*, j^*, \Delta_{i^*,j^*}^*) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$;
– **if** $\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupted}}$: **return** $\perp$;
– $(m_0^*, m_1^*) \leftarrow \mathcal{C}_{\mathsf{msg}}$;
– $C^\dagger \xleftarrow{\$} \overline{\mathsf{O}_{\mathsf{challenge}}}(m_0^*, m_1^*, j^*)$;
– $\mathcal{C}_{\mathsf{msg}}[(j, C^\dagger)] = m_b^*$;
– **return** $C^\dagger$.

Note that $\mathcal{A}$ has no access to $\mathcal{C}_{\mathsf{msg}}$ and thus the use of this set has no impact on the simulation of $\mathsf{O}_{\mathsf{challenge},0}$.

When $b = 0$, then $C^\dagger \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_j, m_0^*)$; otherwise, if $b = 1$, then $C^\dagger \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_j, m_1^*)$. It is now simply the case that $\mathcal{B}$ simulates $\overline{\mathsf{O}_{\mathsf{challenge},b}}$ depending on the value of b. Thus, $\mathcal{B}$ simply outputs $b' = b_{\mathcal{A}}$ where $b_{\mathcal{A}} \leftarrow \mathcal{A}$ is output by $\mathcal{A}$. As a result, let δ be the advantage of $\mathcal{A}$ in distinguishing the scenarios in both games. Then δ is bounded by the advantage that $\mathcal{B}$ has in $\mathsf{IND\text{-}HRA}_{\mathcal{B}}^{\mathcal{PRE},b}$. By the assumption that $\mathcal{PRE}$ satisfies ϵ_1-IND-HRA security, then $\delta < \epsilon_1$ and the lemma follows.

Lemma 9. *The advantage in distinguishing* Game_2 *and* Game_3 *is bounded by* ϵ_2.

Proof. The simulation for $\mathcal{A}$ by $\mathcal{B}$ is the same as in Lemma 7 except in reverse. That is, Game_2 corresponds to the case where $c = 1$ and Game_3 to the case where $c = 0$. The rest of the details are covered by the previous simulation and so we refer the reader to this proof for the full details.

Lemma 10. *The advantage of* $\mathcal{A}$ *in distinguishing the executions of* $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE},b}$ *in* Game_3 *is* 0 *for* $b \xleftarrow{\$} \{0,1\}$.

Proof. In Game_3, the sampling of b is independent of the values that are returned. In particular, the ciphertext $C^\dagger$ returned by $\mathsf{O}_{\mathsf{challenge},b}$ is always a re-encryption of the right input ciphertext C_1^*. Therefore, there is no plausible advantage that $\mathcal{A}$ has to gain in distinguishing the two games and the lemma follows.

To complete the proof of the theorem, we note that the advantage, ϵ, that $\mathcal{A}$ has in distinguishing $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE},b}$ for $b \xleftarrow{\$} \{0,1\}$ is bounded by $2\epsilon_2 + \epsilon_1$. Providing that there is negligible function s.t. $\epsilon_1, \epsilon_2 < \mathsf{negl}(\lambda)$, then we have that $\epsilon < \mathsf{negl}(\lambda)$ and the proof of Theorem 2 is complete.

Theorem 3 (main). *Let* $\mathcal{PRE}$ *be a PRE scheme which is both PKE-CPA-secure and source-hiding. Then* $\mathcal{PRE}$ *also has PCS.*

Proof. It has been shown that PKE-CPA-security and source-hiding imply IND-HRA-security [17, Theorem 6]. This together with Theorem 2 gives us the result. For results on the precise loss of security, see [17].

Existing PRE schemes that satisfy PCS. Using the result of Theorem 2, we can show that a number of existing PRE schemes satisfy PCS immediately. This is advantageous since it shows that PCS is not a vacuous security model in the sense that it is achievable via well-known techniques.

Specifically, any PRE scheme that satisfies PRE-CPA-security along with source-hiding is immediately a scheme that satisfies PCS. Therefore, the schemes of [17, Construction 2, Construction 4, Construction 7.b] all satisfy PCS. Constructions 2 and 4 of [17] are based on assumptions over groups with bilinear maps; whereas Construction 7.b is based on the hardness of the decision learning with errors problem with sub-exponential noise-to-modulus ratio.

6 An Efficient Construction from Lattices

We introduce a natural construction with PCS, based on BV-PRE – the ring-LWE (RLWE) construction given in [30]. Whilst Theorem 3 shows that source-hiding can lead to PCS, the existing constructions with this property [17] make making sub-optimal parameter choices that significantly impact the schemes' practicality. Our construction has PCS but is not source-hiding, implying that source-hiding is not necessary for PCS. This means that our construction can make much better parameter choices in terms of efficiency. Our construction also doesn't rely on strong assumptions or heavy techniques such as obfuscation [11,19]. We also achieve *transparency*, which means decryption is the same regardless of how many times the ciphertext has been re-encrypted, and the cost of decryption does not grow for repeatedly re-encrypted ciphertexts. If extra computation is needed to decrypt a re-encrypted ciphertext, then this may outweigh the benefits of outsourcing re-encryption. This fits better with motivations for outsourcing re-encryption, as heavier computation may go against the reasons for outsourcing re-encryption to begin with.

Our construction makes some adaptations to BV-PRE to fit the workflow of PRE; making use of the key resampling technique of [8] to re-randomise the ciphertext. Any scheme that permits similar re-randomisation can be proven secure using related methods. We begin this section by covering some necessary preliminaries for lattices.

6.1 Lattice preliminaries

We let $q \in \mathbb{Z}$ denote some modulus and n denote a ring dimension. Furthermore, we represent the set of integers modulo q as $\mathbb{Z}_q = \{\lfloor -q/2 \rfloor, \ldots, 0, \ldots, \lfloor q/2 \rfloor\}$ We will be working over power-of-two cyclotomic rings of the form $R_q = \mathbb{Z}_q[x]/(x^n+1)$. Next we discuss the various distributions that will be used. We use the notation $s \xleftarrow{\$} D$ to denote that the element s is sampled according to distribution D. If D is a set, then we assume $s \xleftarrow{\$} D$ means that s is sampled uniformly from the set D. We denote the discrete Gaussian distribution over $\mathbb{Z}_q$ as χ_σ. The distribution χ_σ has its support restricted to $\mathbb{Z}_q$ and a probability mass function proportional to that of a Gaussian distribution with variance σ^2. We sometimes write χ_e where we use a subscript e to denote an "error" distribution, but the underlying variance is still denoted by σ. Slightly abusing notation, we can sample a polynomial $s \xleftarrow{\$} \chi_e$ by sampling each of the coefficients of s according to the distribution χ_e. We say a distribution D is (B, δ)-bounded if $\Pr\left(|x| > B : x \xleftarrow{\$} D\right) \leq \delta$.

Ring-RLWE (RLWE) assumption: Let s be some secret polynomial in R_q. Samples from the $\mathrm{RLWE}_{n,q,\chi_e}(s)$ distribution take the form $(a, b = as + e) \in R_q \times R_q$ where $a \xleftarrow{\$} R_q$, $e \xleftarrow{\$} \chi_e$. Note that χ_e is referred to as the error distribution. The (normal form) $\mathrm{RLWE}_{n,q,\chi_e}$ problem is to distinguish between an oracle that outputs samples from $\mathrm{RLWE}_{n,q,\chi_e}(s)$ where $s \xleftarrow{\$} \chi_e$ and an oracle that outputs

uniform elements in $R_q \times R_q$. The $\mathrm{RLWE}_{n,q,\chi_e}$ assumption states that no PPT algorithm can solve the $\mathrm{RLWE}_{n,q,\chi_e}$ problem with a non-negligible advantage. Note that if we take $\sigma \geq \omega(\log n)$ and $\sigma/q = 1/\mathrm{poly}(n)$, the $\mathrm{RLWE}_{n,q,\chi_e}$ problem is at least as hard as solving standard worst-case lattice problems over ideal lattices up to polynomial approximation factors using quantum algorithms [25].

6.2 Adapting BV-PRE for PCS

The underlying scheme, BV-PRE [30], is based on the BV-encryption scheme [8], which is based on RLWE. This scheme is parameterised by ciphertext modulus q, plaintext modulus $p \geq 2$, ring dimension n, polynomial ring $R_q = \mathbb{Z}_q[n]/\langle x^n + 1\rangle$ and relinearisation window r. BV-PRE is not fully public-key, relying on an additional 'public' key 'pk'$_B$ for the target key s_B to generate update tokens. However, this key together with the token can be used to derive the old secret key. We get around this problem using the key resampling technique ReSample [8] which takes a public key pk_B and outputs a fresh public key pk'_B with the same underlying secret. We also use same relinearisation technique as [30] to reduce error growth. We refer readers interested in the full BV-PRE and re-sampling construction to Appendix C.

$\mathsf{KeyGen}(1^\lambda)$

$a \xleftarrow{\$} \mathcal{U}_q,$
$s, e \xleftarrow{\$} \chi_e$
$b = a \cdot s + pe$
$\mathsf{sk} = s$
$\mathsf{pk} = (a, b)$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk}, m)$

$(a, b) \leftarrow \mathsf{pk}$
$v, e_0, e_1 \xleftarrow{\$} \chi_e$
$c_0 = b \cdot v + pe_0 + m$
$c_1 = a \cdot v + pe_1$
return $c = (c_0, c_1)$

$\mathsf{Dec}(\mathsf{sk}, c)$

$s \leftarrow \mathsf{sk}$
$m' = c_0 - s \cdot c_1 \mod p$
return m'

$\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B)$

for $i \in \{0, 1, \ldots, \lfloor \log_2(q)/r \rfloor\}$:
  $(\beta_i, \theta_i) \xleftarrow{\$} \mathsf{ReSample}(\mathsf{pk}_B)$
  $\gamma_i = \theta_i - \mathsf{sk}_A \cdot (2^r)^i$
$\Delta_{A\to B} = \{(\beta_i, \gamma_i)\}_{i=0}^{\lfloor \log_2(q)/r \rfloor}$
return $\Delta_{A\to B}$

$\mathsf{ReEnc}(\Delta_{A\to B}, \mathsf{pk}_B, c)$

$\{(\beta_i, \gamma_i)\}_{i=0}^{\lfloor \log_2(q)/r \rfloor} \leftarrow \Delta_{A\to B}$
$(c_0, c_1) \leftarrow c$
$(\beta^{\mathrm{proxy}}, \theta^{\mathrm{proxy}}) \leftarrow \mathsf{ReSample}(\mathsf{pk}_B)$
$c'_0 = c_0 + \sum_{i=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(i)} \cdot \gamma_i) + \theta^{\mathrm{proxy}}$
$c'_1 = \sum_{i=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(i)} \cdot \beta_i) + \beta^{\mathrm{proxy}}$
return $c' = (c'_0, c'_1)$

Figure 8: pcBV-PRE. This adapts the BV-PRE scheme [30] to be fully public-key, and minimises computation and bandwidth for the user for the re-encryption process. The key resampling algorithm ReSample is described in Appendix C.

We give our construction, pcBV-PRE, in Figure 8. It builds on BV-PRE in that randomness is also added by the proxy in the ReEnc operation. Recall that this is necessary for a scheme to have PCS, as otherwise an adversary could re-encrypt locally to obtain the same re-encryption. This change has minor implications for how this affects the correctness requirement for multiple re-encryptions over that given in [30]. Note also that pcBV-PRE inherits the IND-CPA-security proven in [30]. This is proven by following the same exact steps from the proof in [30], so we omit considerations of IND-CPA-security for brevity.

6.3 Security proofs

For brevity, we defer the correctness analysis of pcBV-PRE to Appendix C.

Post-Compromise Security. We firstly show that pcBV-PRE is satisfies PCS with a direct proof. In other words, we do not leverage the proof of Theorem 2 to prove PCS via source-hiding security. This is because pcBV-PRE as written does not satisfy source-hiding security, see Section 7.3 for more details.

Theorem 4. pcBV-PRE *has Post-Compromise Security. In other words, for any adversary* $\mathcal{A}$ *to the* PostComp *game,*

$$\left|\Pr\left[\mathsf{PostComp}_{\mathcal{A}}^{0,\mathcal{PRE}}(1^\lambda)=1\right]-\Pr\left[\mathsf{PostComp}_{\mathcal{A}}^{1,\mathcal{PRE}}(1^\lambda)=1\right]\right|\leq\epsilon,$$

for some $\epsilon=\mathsf{negl}(\lambda)$ *under the* $RLWE_{n,q,\chi_e}$ *assumption.*

Proof Overview. The overall structure of the proof is essentially the same as the IND-CPA-security proof of the BV-PRE in [30]. We only give a brief informal argument here and refer the reader to the Appendix C for a full proof. The proof follows a sequence of game hops beginning with the $\mathsf{PostComp}^{b,\mathcal{PRE}}$ security game where $b \xleftarrow{\$} \{0,1\}$. In this game, the adversary is challenged to guess the bit b. Suppose that there are N honest entities that are not corrupted by the adversary. Then we begin by making N game hops. Each game hop replaces:

1. the public key of a single honest entity with a uniform random value.
2. the re-encryption keys created using the honest entity's public key with uniform random values.

In the final game hop, the challenge ciphertext given to the adversary is a uniformly sampled value and thus the adversary has no advantage in this game. This implies that $\mathsf{PostComp}^{0,\mathcal{PRE}}$ and $\mathsf{PostComp}^{1,\mathcal{PRE}}$ are indistinguishable.

IND-CPA security. We secondly show that pcBV-PRE is a IND-CPA-secure PRE scheme. This is a much simpler proof since it follows a similar argument to that of Theorem 4, and the fact that the same security notion was shown in [30, Theorem 2].

Theorem 5. pcBV-PRE *is IND-CPA secure.*

Proof. The proof of this property follows a very similar argument to that of Theorem 4. Firstly, we employ the hybrid argument of Theorem 4 up to the hybrid before the final game hop is made. At this point in the security proof, the public key of all honest entities are replaced with uniform values and similarly for the re-encryption keys. The IND-CPA requirement requires that the challenge ciphertext is encrypted with respect to an honest entity. Therefore, any call to $\mathsf{O}_{\mathsf{ReKeyGen}}$ is answered with a uniform response, and this is the same for responses from $\mathsf{O}_{\mathsf{ReEnc}}$.

In this case, pcBV-PRE is almost identical to the original BV-PRE scheme, since the KeyGen, Enc and Dec algorithms are defined in the exactly the same way. Moreover, the outputs of the additional algorithms are uniform using the hybrid approach above. Therefore, we can leverage the proof of [30, Theorem 2] showing that BV-PRE is IND-CPA-secure. Notice that this is possible since the oracles above can be instantiated with uniform responses, and the other oracles can be handled as in the reduction of [30]. This completes the proof. □

7 From selective to adaptive security

Thus far, we have discussed selective security, where the adversary first corrupts keys, then learns challenges in two distinct stages. In the adaptive model, a single-stage adversary can corrupt secret keys at any point and therefore choose which keys to corrupt as a result of received challenges. As long as the trivial win condition holds, the adversary can adaptively decide which keys should be corrupted depending on the output of other oracle queries.

To lift to this stronger model, we need notions of key privacy. The intuition behind this is that if we can replace some challenge-related queries with counterparts under different keys, and the adversary cannot tell that this has happened, then the advantage related to specific corruptions in the adaptive model is negligible.

7.1 Weak Key Privacy

In this section, we give the formal definitions for the related notions of key privacy. *key privacy* (sometimes called *key anonymity*), was first introduced for PKE in [4]. Informally it means that an adversary is unable to determine which public key was used to encrypt a ciphertext. A strong version for PRE schemes which accounts for update tokens appears in [1] which is referred to as *strong key privacy* in [17], and states that an adversary is unable to distinguish between an update token between two uncorrupted keys and a random element of the token space. Note that this is strictly stronger than key inference where the hidden secret keys must be computed, as is is the concern in *collusion attacks*. We refer the reader to [1] for further details.

In their work relating selective and adaptive security, Fuchsbauer et al. [17] define *weak key privacy* and show that this is sufficient for adaptive security. We now define this formally.

$\mathsf{weakKP}^{b,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda, 1^\kappa)$

$(\mathsf{pk}_0, \mathsf{sk}_0), \ldots (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
for $i \in \{1, \ldots, \kappa\}$:
  $\Delta^{(0)}_{0,1} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_0, \mathsf{pk}_i)$
  $\Delta^{(1)}_{0,1} \leftarrow \mathsf{ReKeyGen}^{\mathsf{wKP}}(\mathsf{pk}_i)$
$b' \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}_0, \ldots, \mathsf{pk}_\kappa, \Delta^{(b)}_{0,1}, \ldots, \Delta^{(b)}_{0,\kappa})$
return b'

$\mathsf{ReKeyGen}^{\mathsf{wKP}}(\mathsf{pk}_i)$

$(\mathsf{pk}'_0, \mathsf{sk}'_0) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$\Delta^{(1)}_{0,i} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}'_0, \mathsf{pk}_i)$
return $\Delta^{(1)}_{0,i}$

Figure 9: The weak key privacy game. Given an update token, an adversary cannot distinguish the source public key.

Definition 8. *A PRE scheme $\mathcal{PRE}$ is said to have* (ϵ, κ)-weak key privacy *if for all* PPT *adversaries $\mathcal{A}$:*

$$\left|\Pr\left[\mathsf{weakKP}^0_{\mathcal{A}}(1^\lambda, 1^\kappa) = 1\right] - \Pr\left[\mathsf{weakKP}^1_{\mathcal{A}}(1^\lambda, 1^\kappa) = 1\right]\right| \leq \epsilon,$$

where $\mathsf{weakKP}^{b,\mathcal{PRE}}_{\mathcal{A}}$ *is defined in Figure 9.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme has κ-weak key privacy*. If this holds for all polynomial κ with respect to the security parameter, then the scheme has* weak key privacy*.*

7.2 Adaptive Post-Compromise Security

In this section, discuss PCS with adaptive key corruptions. In an adaptive game, the adversary is in one stage and has access to all oracles at once, and so is free to choose which keys to corrupt based on previous oracle responses. We give the full adaptive game in Appendix E.

In general, selective security can been shown to imply adaptive security at an exponential loss, where the adversary in the selective game must first guess which keys the adaptive adversary will corrupt. This is known as *complexity leveraging*. A general framework for showing relations between selective and adaptive security for general definitions is given in [20]. Jafargholi et al. achieve this by constructing a variant of the selective game, then using pebbling games and a series of hybrids to the adaptive game to demonstrate the loss in security between each hybrid. Using pebbling games can give a smaller loss of security. A similar approach specific to PRE is used in [17], limited to when the adversary creates directed re-encryption graphs which are acyclic and have only one source node. An even tighter reduction is given which results in quasi-polynomial loss between selective and adaptive HRA security for PRE schemes with ciphertext indistinguishability and weak key privacy, limited to some types of directed re-encryption graph, namely trees and chains.

Selective PRE-CPA security combined with weak key privacy gives adaptive PRE-CPA security, where the loss of security depends on the number of keys κ generated during the game, and the space and time complexities of the $\mathcal{DRG}$.

More precisely, the loss of security is approximately $\tau \cdot \kappa^{\sigma}$, where σ is the maximum number of pebbles and τ is the maximum number of moves for a valid pebbling strategy for a considered class of re-encryption graphs $\mathcal{DRG}$ [17, Theorem 1, Theorem 5]. This means that the loss is exponential for general re-encryption graphs, but quasi-polynomial for trees and chains.

Theorem 6 (main). *Let $\mathcal{PRE}$ be a PRE scheme which is ϵ_1-selectively PRE-CPA secure, ϵ_2-source-hiding and ϵ_3-weakly key private. Then it is ϵ-adaptively PCS secure with a security loss of $\approx \tau \cdot \kappa^{\sigma}$ for directed re-encryption graphs $\mathcal{DRG}$ with degree κ, space complexity σ and time complexity τ, where*

$$\epsilon \leq 2\kappa(\kappa - 1)(Q_E + Q_{RE})Q_{RE} \cdot \epsilon_{SH} + \kappa^{\sigma+\delta+1}(\epsilon_{IND} + 2\tau \cdot \epsilon_3) \qquad (1)$$

restricted to graphs $\mathcal{DRG}$ which are acyclic, have at most κ nodes, degree δ and depth d.

The proof of this follows from Theorem 3 and uses the same strategies as those used to prove [17, Theorem 5].

7.3 Achieving adaptive security of RLWE-based construction

The construction given in Figure 8 is not source-hiding. This is because fresh encryptions will have different noise magnitudes compared with re-encryptions. One method of overcoming this is using a noise 'blurring' approach [10], this was noted by [17, Construction 7.b]. This would involve 'blurring' or 'drowning' the noise in fresh encryptions so that it was distributed in the same way as re-encryptions. To do this, we would modify the ciphertext pairs in the Enc and ReEnc algorithms so that we would add fresh noise to them. This fresh noise would be taken from an error distribution such that it 'blurred' out all the old noise that was introduced from the encryption and re-encryption operations.

The advantage of doing this, is that on decryption the noise will not reveal anything about the method of encryption/re-encryption that was used, or the keys the ciphertext was previously encrypted under. This is vital since in the source-hiding security property, the adversary receives all of the available decryption keys and thus can decrypt any ciphertext that they want.

With this in mind, there are a number of reasons why proving source-hiding is actually a hindrance. Using the same approach as [17] requires at least a sub-exponential noise-to-modulus ratio which considerably harms performance and security. This is because the LWE assumption (and respectively RLWE in our case) that is used becomes much stronger, since the approximation factors of the related ideal lattice problems become sub-exponential rather than polynomial in n. In particular, there are quantum polynomial time algorithms solving ideal lattice problems with approximation factor $\exp(\tilde{O}(\sqrt{n}))$ [5, 14, 15] providing evidence that we cannot simply use an arbitrary noise-to-modulus ratio while retaining the same security guarantees. Moreover, the increase in modulus that is required makes standard operations much slower. As a by-product, the scheme of [17] allows only a constant number of re-encryptions.

The intention with our construction was to give a practical PCS PRE scheme with minimal restrictions and from weak assumptions. Since our construction is very close to the original [30] definition, we achieve this goal. Since BV-PRE is comparatively fast (see [30] for exact figures) and our construction only adds extra sampling, loss of efficiency is minimal. If we wanted to incorporate source-hiding, this would result in a scheme that was impractical, based on much stronger assumptions and also heavily restricted in the number of re-encryptions that can occur. Therefore, we choose not give an explicit formulation.

We prove that our scheme has weak key privacy in Appendix F. Therefore, it should be noted that a source-hiding version of our scheme would achieve adaptive security (for the restricted graphs of [17]) by the proof of Theorem 2 and the fact that our scheme is IND-CPA secure 5. This result was explicitly shown in [17, Lemma 7].

8 Conclusions and Future work

In this paper, we have presented the strongest notion of Post-Compromise Security for PRE to date. By strongest, we mean that existing PCS notions are implied by our notion of security, and separating examples showing that the opposite implication does not follow exist. We have also shown that PCS can be achieved via a number of existing PRE security notions which immediately shows that there are existing PRE schemes that satisfy PCS [17]. Finally, we give a practical construction of a PCS secure PRE scheme with transparency which is based on lattices and discussed the possibility of achieving adaptive security for restricted re-encryption graphs. We leave as future work the possibility of proving tighter bounds between security notions, and further investigating the relationship between selective and adaptive security for more generic graphs.

Acknowledgements

Special thanks to Katriel Cohn-Gordon for his help in motivating this work.

References

1. Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-private proxy re-encryption. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2009.
2. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.
3. Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management part 1: General (revision 3). *NIST special publication*, 800(57):1–147, 2012.

4. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.
5. Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 893–902. SIAM, 2016.
6. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.
7. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013.
8. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.
9. Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 185–194. ACM, 2007.
10. Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2014.
11. Nishanth Chandran, Melissa Chase, and Vinod Vaikuntanathan. Functional re-encryption and collusion-resistant obfuscation. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 404–421. Springer, 2012.
12. Aloni Cohen. What about bob? the inadequacy of CPA security for proxy reencryption. *IACR Cryptology ePrint Archive*, 2017:785, 2017.
13. Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. On post-compromise security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 164–178. IEEE Computer Society, 2016.

14. Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 559–585. Springer, 2016.
15. Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-svp. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 324–348, 2017.
16. Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. *IACR Cryptology ePrint Archive*, 2017:527, 2017.
17. Georg Fuchsbauer, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Adaptively secure proxy re-encryption. *IACR Cryptology ePrint Archive*, 2018:426, 2018.
18. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.
19. Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 233–252. Springer, 2007.
20. Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 133–163. Springer, 2017.
21. Ela Lee. Improved security notions for proxy re-encryption to enforce access control. Cryptology ePrint Archive, Report 2017/824, 2017. `http://eprint.iacr.org/2017/824`.
22. Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 685–716. Springer, 2018.
23. Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In Ronald Cramer, editor, *Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*, volume 4939 of *Lecture Notes in Computer Science*, pages 360–379. Springer, 2008.
24. Adriana Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. Cryptology ePrint Archive, Report 2013/094, 2013. `https://eprint.iacr.org/2013/094`.

25. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.
26. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.
27. Steven Myers and Adam Shull. Efficient hybrid proxy re-encryption for practical revocation and key rotation. *IACR Cryptology ePrint Archive*, 2017:833, 2017.
28. OWASP. Cryptographic storage cheat sheet. `https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet`, 2018. Accessed 9th October 2018.
29. PCI Security Standards Council 2018. Payment Card Industry (PCI) data security standard (version 3.2.1). 2018.
30. Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.*, 20(4):14:1–14:31, 2017.

A Definitions for symmetric PRE

Here we give definitions for the symmetric setting. We leave out some security games where the change is analogous, but explicitly give those which contains lists to clarify how those lists are updated.

Definition 9. *A* symmetric Proxy Re-Encryption (symPRE) *scheme consists of the following algorithms:*

- $\mathsf{sym.Setup}(1^\lambda) \to$ *params: Outputs a set of public parameters, including the message space and ciphertext space. Note that params is input to every subsequent algorithm, but we leave it out for compactness of notation. We often omit the* Setup *algorithm for the same reason.*
- $\mathsf{sym.KeyGen}(1^\lambda) \to k$*: Generates a secret key*
- $\mathsf{sym.Enc}(k, m, \ell) \to C$*: Encrypts a message* m *using a key* k*, producing a ciphertext at level* ℓ*. We often leave out* ℓ *for compactness.*
- $\mathsf{sym.Dec}(k, C) \to m' \cup \perp$*: Decrypts a ciphertext* C *to produce either an element of the message space* m' *or the error symbol*
- $\mathsf{sym.ReKeyGen}(k_i, k_j) \to \Delta_{i,j} \cup \perp$*: Takes current key* k_i *and next key* k_j *and outputs an* update token $\Delta_{i,j}$*, or* $\perp$ *when* $i = j$*. This last condition is often left out of constructions for compactness.*
- $\mathsf{sym.ReEnc}(\Delta_{i,j}, C) \to C'$*: Takes a ciphertext* C *under* k_i *and outputs a new ciphertext* C' *under* k_j*.*

A symmetric PRE scheme is correct *if for all* $m \in \mathcal{M}, k \xleftarrow{\$} \mathsf{sym.KeyGen}(1^\lambda)$*:*

$$\mathsf{sym.Dec}(k, \mathsf{Enc}(k, m)) \to m$$

and if for all $C \in \mathcal{C}$ *such that* $\mathsf{sym.Dec}(k_i, C) \to m$*:*

$$\mathsf{sym.Dec}(k_j, \mathsf{sym.ReEnc}(\Delta_{i,j}, C)) \to m$$

where $k_i, k_j \xleftarrow{\$} \mathsf{sym.KeyGen}(1^\lambda)$ *and* $\Delta_{i,j} \leftarrow \mathsf{sym.ReKeyGen}(k_i, k_j)$*.*

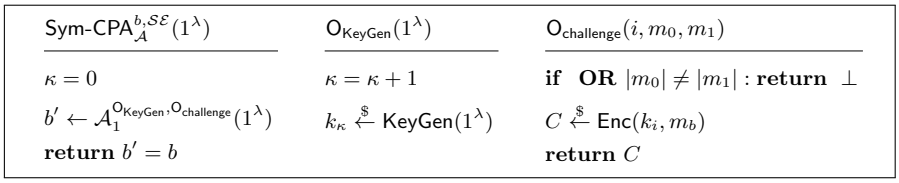

Figure 10: The Sym-CPA, the symmetric variant of the PKE-CPA game.

Definition 10. *An encryption scheme $\mathcal{SE}$ is* ϵ-Indistinguishable against Chosen Plaintext Attacks (ϵ-sym-CPA-secure) *if for all* PPT *adversaries $\mathcal{A}$:*

$$\left|\Pr\left[\mathsf{Sym\text{-}CPA}^{0,\mathcal{SE}}_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Sym\text{-}CPA}^{1,\mathcal{SE}}_{\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \epsilon,$$

where Sym-CPA *is defined in Figure 10. If ϵ is negligible as parameterised by the security parameter λ, then we say the scheme is*Indistinguishable against Chosen Plaintext Attacks (sym-CPA-secure)*.*
A symmetric PRE scheme $\mathsf{sym}\mathcal{PRE}$ *is* (ϵ-sym-CPA-secure) *if the encryption scheme given by* $\mathsf{sym}\mathcal{PKE} = \{\mathsf{sym}\mathcal{PRE}.\mathsf{KeyGen}, \mathsf{sym}\mathcal{PRE}.\mathsf{Enc}, \mathsf{sym}\mathcal{PRE}.\mathsf{Dec}\}$ *is* ϵ-Sym-CPA *secure.*

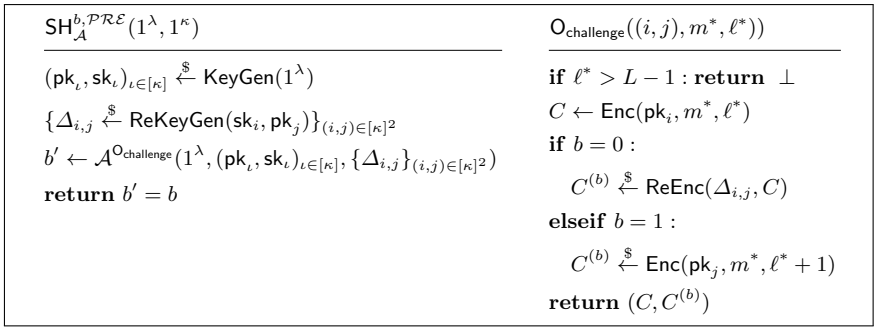

Figure 11: Experiments for the symmetric source-hiding property. L is the number of times a ciphertext can be re-encrypted without breaking the correctness conditions.

Definition 11. *A symmetric PRE scheme $\int\mathcal{PRE}$ is said to be* ϵ-post-compromise secure (ϵ-PCS) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left|\Pr\left[\mathsf{sSymPostComp}^{0,\int\mathcal{PRE}}_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{sSymPostComp}^{1,\int\mathcal{PRE}}_{\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \epsilon,$$

where $\mathsf{sSymPostComp}\mathcal{A}^{b,\mathcal{PRE}}$ *is defined in Figure 13.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme is post-compromise secure (PCS)*.*

$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda)$

$\kappa = \kappa + 1$

$k_\kappa \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$

$\mathcal{DRG}.\mathsf{add}\ (v_\kappa)$

$\mathsf{O}_{\mathsf{Corrupt}}(i)$

$\mathcal{K}_{\mathsf{corrupted}}.\mathsf{add}\ (k_i)$

return k_i

$\mathsf{O}_{\mathsf{Enc}}(i, m)$

$C \xleftarrow{\$} \mathsf{Enc}(k_i, m)$

$\boxed{\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (i, C)}$

$\boxed{\mathcal{C}_{\mathsf{msg}}[(i, C)] = m}$

return C

$\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$

$\Delta_{i,j} \xleftarrow{\$} \mathsf{ReKeyGen}(k_i, k_j)$

$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\ (i, j, \Delta_{i,j})$

$\mathcal{DRG}.\mathsf{add}\ (\vec{e}_{i,j})$

return $\Delta_{i,j}$

Figure 12: Common oracles used in security games for symmetric PRE.

$\mathsf{sSymPostComp}_{\mathcal{A}}^{b,\mathcal{PRE}}(1^\lambda)$

$\mathcal{K}_{\mathsf{chal}}, \mathcal{K}_{\mathsf{corrupted}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}}, \mathcal{C}_{\mathsf{msg}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{DRG} = \emptyset$

$\kappa = 0, \mathsf{called} = \mathsf{false}$

$state \leftarrow \mathcal{A}_0^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReEnc}}, \mathsf{O}_{\mathsf{Corrupt}}}(1^\lambda)$

$b' \leftarrow \mathcal{A}_1^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{O}_{\mathsf{ReEnc}}, \mathsf{O}_{\mathsf{challenge}}}(1^\lambda, state)$

$\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$

if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupted}} \neq \emptyset$: **return** $\perp$

return $b' = b$

$\mathsf{O}_{\mathsf{ReEnc}}(C, i, j, [\Delta_{i,j}])$

if $\Delta_{i,j}$ given :

  if $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$

else : $\Delta_{i,j} \leftarrow \mathsf{ReKeyGen}(k_i, k_j)$

if $(i, C) \notin \mathcal{C}_{\mathsf{honest}}$: **return** $\perp$

$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C)$

$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C')$

$\mathcal{C}_{\mathsf{msg}}[(j, C')] = \mathcal{C}_{\mathsf{msg}}[(i, C)]$

if $(i, C) \in \mathcal{C}_{\mathsf{chal}}$:

  $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C'), \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (k_j)$

return C'

$\mathsf{O}_{\mathsf{challenge}}(C_0, C_1, i, j, \Delta_{i,j})$

if $|C_0| \neq |C_1|$ **OR** $\mathsf{called} = \mathsf{true}$: **return** $\perp$

if $(i, C_0), (i, C_1) \notin \mathcal{C}_{\mathsf{honest}}$ **OR** $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$

$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C_b)$

$\mathcal{C}_{\mathsf{msg}}[(j, C')] = \mathcal{C}_{\mathsf{msg}}[(i, C_b)]$

$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C'), \mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C'), \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (k_j)$

$\mathsf{called} \leftarrow \mathsf{true}$

return C'

Figure 13: The symmetric variant of PostComp game.

We give the equivalent lemma to Lemma 5 for symmetic PRE.

Lemma 11. *IND-UPD* $\not\Longrightarrow$ *symmetric PCS.*

Proof. We present the following counterexample for symmetric PRE, showing that IND-UPD $\not\Longrightarrow$ symmetric PCS. RISE [22] is proven to be IND-UPD. Here we adapt RISE for general proxy re-encryption as this fits more with our notation, but we observe that the original construction defined as an updatable encryption scheme is also sufficient for the proof.

- $\mathsf{RISE}'.\mathsf{KeyGen}(1^\lambda) : x \xleftarrow{\$} \mathbb{Z}_q^*, (\mathsf{pk}, \mathsf{sk}) =, (x, g^x)$
- $\mathsf{RISE}'.\mathsf{Enc}(\mathsf{pk}, m) : r \xleftarrow{\$} \mathbb{Z}_q^*, C \leftarrow (\mathsf{pk}^r, g^r \cdot m)$

– $\mathsf{RISE}'.\mathsf{Dec}(\mathsf{sk}, m) : m' \leftarrow C_1 \cdot C_0^{-1/\mathsf{sk}}$
– $\mathsf{RISE}'.\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j) : \Delta_{i,j} = (\mathsf{sk}_j/\mathsf{sk}_i, \mathsf{pk}_j) = (x_j/x_i, g^{x_j})$
– $\mathsf{RISE}'.\mathsf{ReEnc}(\Delta_{i,j}, C) : (\Delta, y') = \Delta_{i,j}, r' \xleftarrow{\$} \mathbb{Z}_q^*, C' \leftarrow (C_0^{\Delta} \cdot y'^{r'}, C_1 \cdot g^{r'})$

Given $\Delta_{i,j} = (x_j/x_i, g^{x_j})$, and x_i, $\mathcal{A}$ can compute x_j and use this to decrypt the challenge ciphertext. This is because RISE is not unidirectional it is not compromise secure by Lemma 2.

B Asymmetric IND-UPD

IND-UPD [22, definition 3] is a post-compromise security notion for updatable encryption schemes, which are a variant of symmetric PCS schemes. Here we adapt IND-UPD for the public-key setting.

Definition 12. *A PRE scheme $\mathcal{PRE}$ is said to be* (selectively) ϵ-pkIND-UPD-secure *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left|\Pr\left[\mathsf{pkIND\text{-}UPD}^0_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{pkIND\text{-}UPD}^1_{\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \epsilon,$$

where $\mathsf{pkIND\text{-}UPD}\mathcal{A}^{b,\mathcal{PRE}}$ *is defined in Figure 14.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme is (selectively) pkIND-UPD-secure.

C BV-PRE

Here we give the BV-PRE scheme [30] as well as the full proof of Theorem 4. This scheme is not fully public-key as it uses additional 'public' key 'pk'_B for the target key, which cannot be made public without exposing sk_A to anyone with the update token.

Key Resampling. The key resampling technique [8], can be found in Figure 16. We use this in our construction in Figure 8 to obtain PCS.

C.1 Correctness of our modified BV-PRE scheme

Much of the analysis used to argue correctness from [30] holds for our modified scheme in Figure 8. Therefore, we keep the discussion fairly brief and refer to [30] for full details.

Recall from [30] that on input ciphertext (c_0, c_1) and secret key s, the decryption algorithm computes $c_0 - c_1 s$ and then performs a reduction modulo p. In the case that (c_0, c_1) is a ciphertext produced by the Enc algorithm, we have that $c_0 - c_1 s = p(ev + e_0 + e_1 s) + m \bmod q$ and that decryption is successful when $p(ev + e_0 + e_1 s) + m$ (i.e. the error plus the message) does not wrap around modulo q. However, the correctness condition of decryption is different when considering ciphertexts output by the ReEnc algorithm. To see how, let

$\mathsf{pkIND\text{-}UPD}^{b,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$

$\mathcal{K}_{\mathsf{corrupted}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{K}_{\mathsf{chal}}, \mathcal{DRG} = \emptyset$
$\kappa = 0, c = \perp$
$state \leftarrow \mathcal{A}_0^{\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}^{\mathsf{pkIU}}_{\mathsf{LearnTok}}, O^{\mathsf{pkIU}}_{\mathsf{ReEnc}}}(1^\lambda)$
$b' \leftarrow \mathcal{A}_1^{\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}^{\mathsf{pkIU}}_{\mathsf{LearnChal}}, O^{\mathsf{pkIU}}_{\mathsf{ReEnc}}, \mathsf{O}^{\mathsf{pkIU}}_{\mathsf{challenge}}}(1^\lambda, state)$
$\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupted}} \neq \emptyset$: **return** 0
return b'

$\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{KeyGen}}(1^\lambda)$

$\kappa = \kappa + 1$
$\mathcal{DRG}.\mathsf{add}\ (v_\kappa)$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
if $\kappa > 1$:
  $\Delta_{\kappa-1,\kappa} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_{\kappa-1}, \mathsf{pk}_\kappa)$
if $c \neq \perp$:
  $C^*_\kappa \leftarrow \mathsf{ReEnc}(\Delta_{\kappa-1,\kappa}, C^*_{\kappa-1})$
return pk_κ

$\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{LearnTok}}(i)$

if $i = c$: **return** $\perp$
if $\mathcal{PRE}$ is unidirectional :
  $\mathcal{DRG}.\mathsf{add}\ (\vec{e}_{i-1,i})$
elseif $\mathcal{PRE}$ is bidirectional :
  $\mathcal{DRG}.\mathsf{add}\ (e_{i,i-1})$*(undirected edge)*
return $\Delta_{i-1,i}$

$\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{LearnChal}}()$

$\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_\kappa)$
return C^*_κ

$\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{ReEnc}}(C)$

if $(\kappa - 1, C) \notin \mathcal{C}_{\mathsf{honest}}$: **return** $\perp$
if ReEnc is deterministic :
  if $C = \bar{C}_0$ **OR** $C = \bar{C}_1$: **return** $\perp$
$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{\kappa-1,\kappa}, C)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (\kappa, C')$
return C'

$\mathsf{O}^{\mathsf{pkIU}}_{\mathsf{challenge}}(C_0, C_1)$

if $c \neq \perp$ **OR** $|C_0| \neq |C_1|$: **return** $\perp$
if $(\kappa, C_0), (\kappa, C_1) \notin \mathcal{C}_{\mathsf{honest}}$: **return** $\perp$
$\kappa = \kappa + 1, \mathcal{DRG}.\mathsf{add}\ (v_\kappa)$
$c = \kappa, \bar{C}_0 = C_0, \bar{C}_1 = C_1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$\Delta_{\kappa-1,\kappa} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_{\kappa-1}, \mathsf{pk}_\kappa)$
$C^*_\kappa \leftarrow \mathsf{ReEnc}(\Delta_{\kappa-1,\kappa}, C_b)$
$\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_\kappa)$
return $(\mathsf{pk}_\kappa, C^*_\kappa)$

Figure 14: The pkIND-UPD game, based on IND-UPD [22] adapted to the public-key setting.

$(c'_0, c'_1) \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{A\to B}, (c_0, c_1))$. Using notation consistent with Figure 16, we have that

$$\begin{aligned} c'_0 - c'_1 s_B &= c_0 + \sum_{i=0}^{\lfloor \log(q)/r \rfloor} c_1^{(i)} \left(\theta_i - s_A (2^r)^i - s_B \beta_i\right) + (\theta^{\text{proxy}} - \beta^{\text{proxy}} s_B) \\ &= c_0 - c_1 s_A + \sum_{i=0}^{\lfloor \log(q)/r \rfloor} c_1^{(i)} \left(\theta_i - s_B \beta_i\right) + (\theta^{\text{proxy}} - \beta^{\text{proxy}} s_B) \\ &= c_0 - c_1 s_A + \sum_{i=0}^{\lfloor \log(q)/r \rfloor} c_1^{(i)} \cdot p(e v_i + e'_i s_B + e''_i) + p(e^{\text{proxy}} v^{\text{proxy}} + e'^{\text{proxy}} s_B + e''^{\text{proxy}}), \end{aligned}$$

$\mathsf{Setup}(1^\lambda)$

Choose positive integers q, n
Choose plaintext modulus p
Choose key switching window r
return $pp = (q, n, p.r)$

$\mathsf{KeyGen}(1^\lambda)$

$a \xleftarrow{\$} \mathcal{U}_q$
$s, e \xleftarrow{\$} \chi_e$
$b = a \cdot s + pe$
$\mathsf{sk} = s, \mathsf{pk} = (a, b)$
return $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Enc}(\mathsf{pk}, m)$

$(a, b) \leftarrow \mathsf{pk}$
$v, e_0, e_1 \xleftarrow{\$} \chi_e$
$c_0 = b \cdot v + pe_0 + m$
$c_1 = a \cdot v + pe_1$
return $c = (c_0, c_1)$

$\mathsf{Dec}(\mathsf{sk}, c)$

$s \leftarrow \mathsf{sk}, (c_0, c_1) \leftarrow c$
$m' = c_0 - s \cdot c_1 \mod p$
return m'

$\mathsf{Preprocess}(1^\lambda, \mathsf{sk}_B)$

$s_b \leftarrow \mathsf{sk}_b$
for $i \in \{0, 1, \ldots, \lfloor \log_2(q)/r \rfloor\}$:
  $\beta_i \xleftarrow{\$} \mathcal{U}_q$
  $e_i \xleftarrow{\$} \chi_e$
  $\theta_i = \beta_i \cdot s_B + pe_i$
return $\text{`}\mathsf{pk}\text{'}_B = \{(\beta_i, \theta_i)\}_{i=0}^{\lfloor \log_2(q)/r \rfloor}$

$\mathsf{ReKeyGen}(\mathsf{sk}_A, \text{`}\mathsf{pk}\text{'}_B)$

$s_A \leftarrow \mathsf{sk}_A, \{(\beta_i, \theta_i)\}_{i=0}^{\lfloor \log_2(q)/r \rfloor} \leftarrow \text{`}\mathsf{pk}\text{'}_B$
for $i \in \{0, 1, \ldots, \lfloor \log_2(q)/r \rfloor\}$:
  $\gamma_i = \theta_i - s_A \cdot (2^r)^i$
$\Delta_{A \to B} = \{(\beta_i, \gamma_i)\}_{i=0}^{\lfloor \log_2(q)/r \rfloor}$
return $\Delta_{A,B}$

$\mathsf{ReEnc}(\Delta_{A,B}, c)$

$\{(\beta_i, \gamma_i)\}_{i=0}^{\lfloor \log_2(q)/r \rfloor} \leftarrow \Delta_{A,B}$
$(c_0, c_1) \leftarrow c$
$c_0' = c_0 + \sum_{i=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(i)} \cdot \gamma_i)$
$c_1' = \sum_{i=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(i)} \cdot \beta_i)$
return $c' = (c_0', c_1')$

Figure 15: BV-PRE [30] where $\mathcal{U}_q$ is the discrete uniform distribution over R_q and χ_e is a B_e-bounded discrete Gaussian error distribution.

where all arithmetic is done over the integers modulo q. This shows that the error grows by an additive term of pE on each re-encryption where

$$E = \sum_{i=0}^{\lfloor \log(q)/r \rfloor} c_1^{(i)} \cdot (ev_i + e_i' s_B + e_i'') + (e^{\text{proxy}} v^{\text{proxy}} + e'^{\text{proxy}} s_B + e''^{\text{proxy}}).$$

Assume that χ_e is (B, δ)-bounded for some small δ (we quantify these values later). Then we have that the noise grows by at most

$$p||E||_\infty \leq pn(2^r - 1)(\lfloor \log(q)/r \rfloor + 1)(2B^2 n + B) + p(2B^2 n + B) =: G(n, q, r, B) \tag{2}$$

on each re-encryption taking into consideration the multiplication of degree $n - 1$ polynomials. Therefore, after ℓ re-encryptions, the noise has grown by an additive term of size ℓG. Therefore, the condition for successful decryption after ℓ re-encryptions is

$$p(2B^n + B) + \ell G(n, q, r, B) + p/2 \leq q/2 \tag{3}$$

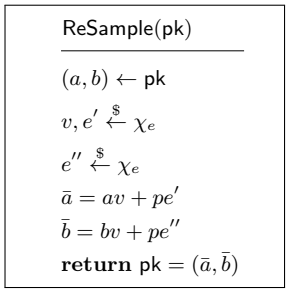

Figure 16: Key resampling technique [8].

where G is define in Equation (2). Note that if our error distribution has $\sigma > \omega(\log n)$, then we can set $B = \sigma\sqrt{n}$ and $\delta = 2^{-n+1}$ [24, 26]. In order to choose parameters of the system, one needs to ensure that Equation 3 is satisfied. Note that this analysis is a worst-case one. It is shown in [30] that the central limit theorem can be used to essentially alter the form of B and change all occurrences of n into occurrences of $\sqrt{n}$ in Equation (3) by allowing for a tunable failure probability. We omit this more practical method of correctness analysis for brevity.

C.2 Full proof of Theorem 4

Proof. Let N be the number of honest parties and M be the total number of parties where M is polynomially bounded. First of all, we label the nodes in our $\mathcal{DRG}$ using a topological ordering (from sinks to sources) such that there are no edges from i to k if $i < k$. In doing so, we have that the honest and corrupt entities have labels in the set $\mathit{\Gamma}_H = \{1, \ldots, N\}$ and $\mathit{\Gamma}_C = \{N+1, \ldots, M\}$ respectively. For $i = 1, \ldots, M$, we denote the secret and public keys of node i as

$$\mathsf{sk}_i = s_i, \quad \mathsf{pk}_i = (a_i, b_i = a_i \cdot s_i + e_i) \tag{4}$$

where $a_i \xleftarrow{\$} R_q$ and $s_i, e_i \xleftarrow{\$} \chi_e$. Also, for $j = 1, \ldots, Q_i$ where Q_i is a polynomial bounding the number of queries to $\mathsf{ReSample}(\mathsf{pk}_i)$, we denote the output of the j^{th} call of the form $\mathsf{ReSample}(\mathsf{pk}_i)$ as the pair

$$a'_{i,j} = a_i v_{i,j} + pe'_{i,j}, \quad b'_{i,j} = b_i v_{i,j} + pe''_{i,j} \tag{5}$$

where $v_{i,j}, e'_{i,j}, e''_{i,j} \xleftarrow{\$} \chi_e$. We now state our main hybrid games between the adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

- Game_0: Run the $\mathsf{PostComp}^{b,\mathcal{PRE}}$ security game where the bit $b \xleftarrow{\$} \{0, 1\}$ uniformly. Denoting the adversary $\mathcal{A}$'s output in the $\mathsf{PostComp}^{b,\mathcal{PRE}}$ as b', the output of Game_0 is 1 if $b' = b$ and 0 otherwise.
- Game_i for $i = 1, \ldots, N$: Same as Game_{i-1} except that the challenger $\mathcal{C}$ replaces $\mathsf{sk}_i, \mathsf{pk}_i$ and all terms $a'_{i,\cdot}, b'_{i,\cdot}$ in (5) by random uniform values.

– Game_{N+1}: $\mathcal{C}$ behaves the same as Game_N except that all challenge ciphertexts output by the challenge oracle are replaced by uniform random values.

Note that Game_0 challenges an adversary to distinguish between the cases when it plays either $\mathsf{PostComp}^{0,\mathcal{PRE}}(1^\lambda)$ and $\mathsf{PostComp}^{1,\mathcal{PRE}}(1^\lambda)$. Therefore, we aim to show that there is no PPT adversary that wins Game_0 with probability a non-negligible amount more than $1/2$. The adversary information theoretically has no advantage in Game_{N+1} i.e.

$$\Pr\left[1 \xleftarrow{\$} \mathsf{Game}_{N+1}(\mathcal{A})\right] - 1/2 = 0.$$

This means that we can prove security by showing that Game_0 is computationally indistinguishable from Game_{N+1} according to the RLWE assumption. We do so by proving the following two lemmas:

Lemma 12. *Game_{i-1} is computationally indistinguishable from Game_i for $i = 1, \ldots, N$ under the RLWE assumption. In particular, let Q_i be the number of times $\mathsf{ReSample}(\mathsf{pk}_i)$ is called during the PostComp game. Then for any PPT adversary $\mathcal{A}$, there exists a RLWE distinguisher $\mathcal{D}$ such that*

$$\left|\Pr\left[1 \xleftarrow{\$} \mathsf{Game}_{i-1}^{k-1}(\mathcal{A})\right] - \Pr\left[1 \xleftarrow{\$} \mathsf{Game}_{i-1}^{k}(\mathcal{A})\right]\right| \leq (1+Q_i)\mathbf{Adv}RLWE_{\phi,q,\chi_e}(\mathcal{D}).$$

Proof. Let Q_i be the polynomially bounded number of times that $\mathsf{ReSample}(\mathsf{pk}_i)$ is called. We define a sequence of sub-hybrids between Game_{i-1} and Game_i:

– Game_{i-1}^0: The same as Game_{i-1} apart from the fact that the challenger replaces the public key pk_i with a uniform random value.
– Game_{i-1}^k for $k = 1, \ldots, P_i$: The same as $\mathsf{Game}_{i-1}^{k-1}$ apart from that the k^{th} call to $\mathsf{ReSample}(\mathsf{pk}_i)$ is replaced by a uniform random value.

Firstly, since node i is not corrupted by the PostComp adversary $\mathcal{A}$, a RLWE adversary $\mathcal{D}$ can embed a RLWE challenge into pk_i to simulate Game_{i-1} or Game_{i-1}^0 depending on whether its input challenge was uniform or not. In particular, since all calls to the resample algorithm are uniform for public keys associated to nodes with label $\leq i-1$, the ReKeyGen oracle outputs uniform random values when $\mathcal{A}$ asks for a re-encryption key with source node i. Therefore, $\mathcal{D}$ simply returns uniform values on these particular ReKeyGen queries. If Game_{i-1} and Game_{i-1}^0 are distinguishable, then $\mathcal{D}$ can solve the RLWE problem. Therefore,

$$\left|\Pr\left[1 \xleftarrow{\$} \mathsf{Game}_{i-1}(\mathcal{A})\right] - \Pr\left[1 \xleftarrow{\$} \mathsf{Game}_{i-1}^0(\mathcal{A})\right]\right| \leq \mathbf{Adv}RLWE_{\phi,q,\chi_e}(\mathcal{D}).$$

Secondly, a RLWE adversary $\mathcal{D}$ can simulate either $\mathsf{Game}_{i-1}^{k-1}$ or Game_{i-1}^k by performing the following:

1. Query the RLWE oracle twice obtain $(a, d_1), (b, d_2)$ and set $\mathsf{pk}_i = (a, b)$.
2. Answer the k^{th} instance of the query $\mathsf{ReSample}(\mathsf{pk}_i)$ as the pair (d_1, d_2)
3. Answer all other queries as in $\mathsf{Game}_{i-1}^{k-1}$

Once again, if $\mathsf{Game}_{i-1}^{k-1}$ and Game_{i-1}^{k} are distinguishable, then $\mathcal{D}$ can solve the RLWE problem. Therefore,

$$\left|\Pr\left[1 \xleftarrow{\$} \mathsf{Game}_{i-1}^{k-1}(\mathcal{A})\right] - \Pr\left[1 \xleftarrow{\$} \mathsf{Game}_{i-1}^{k}(\mathcal{A})\right]\right| \leq \mathbf{Adv}RLWE_{\phi,q,\chi_e}(\mathcal{D}).$$

Using the triangle inequality implies that

$$\left|\Pr\left[\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] - \frac{1}{2}\right| \leq \epsilon\left(\rho \cdot Q_{rk} + Q_{re} + N\right)) \cdot \mathbf{Adv}RLWE_{\phi,q,\chi_e}(\mathcal{D}).$$

and applying the RLWE assumption completes the proof of this lemma. □

Lemma 13. *For any adversary $\mathcal{A}$,* $\Pr\left[1 \xleftarrow{\$} \mathsf{Game}_N(\mathcal{A})\right] = \Pr\left[1 \xleftarrow{\$} \mathsf{Game}_{N+1}(\mathcal{A})\right]$.

Proof. In both Game_N and Game_{N+1}, the adversary submits two ciphertexts $C_0 = (c_{0,0}, c_{0,1})$ and $C_1 = (c_{1,0}, c_{1,1})$ when it queries the challenge oracle. In Game_N, the challenger responds by setting $c_0 = c_{0(b)}$ and $c_1 = c_{1(b)}$ for uniformly chosen $b \xleftarrow{\$} \{0,1\}$ and returning the challenge ciphertext given by

$$c_0' = c_0 + \theta^* + \sum_{i=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(i)} \cdot \gamma_i), \quad c_1' = \beta^* + \sum_{i=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(i)} \cdot \beta_i),$$

where $\sum_{i=0}^{\lfloor \log_2(q)/r \rfloor} 2^{ri} c_1^{(i)} = c_1$. In particular, β^* and θ^* are the result of calling $\mathsf{ReSample}()$ in Game_N on an honest public key and are therefore uniform random values that are used once and never revealed to $\mathcal{A}$. Therefore, c_0' and c_1' are independent uniform random values in Game_N which is the exact case in Game_{N+1}. This argument holds for each of $\mathcal{A}$'s challenge oracle queries. This concludes the proof of this lemma. □

After noting that $\sum_{i=1}^{N} Q_i = \rho Q_{rk} + Q_{re}$ and using the triangle inequality, we see that for all PPT adversaries $\mathcal{A}$:

$$\left|\Pr\left[1 \xleftarrow{\$} \mathsf{Game}_0(\mathcal{A})\right] - \frac{1}{2}\right| \leq \epsilon,$$

for negligible ϵ according to the RLWE assumption. This inequality implies that the games $\mathsf{PostComp}^{0,\mathcal{PRE}}$ and $\mathsf{PostComp}^{1,\mathcal{PRE}}$ are ϵ-indistinguishable as required. This concludes the proof of the theorem. □

D Adaptive HRA security

Here we give an extension of selective IND-HRA security as described in Section 3.3, which allows the adversary to adaptively corrupt keys in response after receiving challenges.

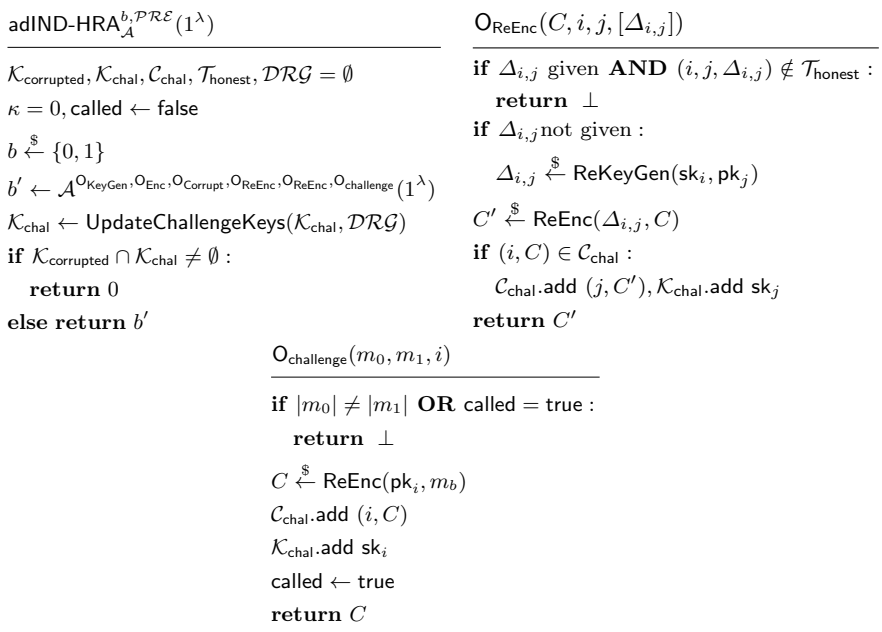

Figure 17: The adIND-HRA game. Like the HRA model [12], it allows re-encryptions of non-challenge ciphertexts to compromised keys using $\mathsf{O}_{\mathsf{ReEnc}}$. It allows adaptive corruption of keys subject to the trivial win condition that no key which a challenge ciphertext has been learned under can be corrupted.

Definition 13. *A PRE scheme $\mathcal{PRE}$ is said to be* ϵ-Adaptively Indistinguishable Honest Re-encryption Plaintext Attacks-secure (ϵ-adIND-HRA-secure) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left|\Pr\left[\mathsf{adIND\text{-}HRA}^{0,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{adIND\text{-}HRA}^{1,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \epsilon, \quad (6)$$

where $\mathsf{adIND\text{-}HRA}^{\mathcal{PRE}}_{\mathcal{A}}$ *is given in Figure 17. If ϵ is negligible as parameterised by the security parameter, then we say the scheme is* Adaptively Indistinguishable Honest Re-encryption Plaintext Attacks-secure (adIND-HRA-secure).

In the adIND-HRA game (Figure 17), $\mathcal{A}$ can adaptively corrupt keys subject to the trivial win condition that they cannot have corrupted a key with which a challenge ciphertext can be decrypted. In order to keep track of the status of keys, the lists $\mathcal{C}_{\mathsf{chal}}, \mathcal{K}_{\mathsf{chal}}$ and $\mathcal{K}_{\mathsf{corrupted}}$ are used as well as a directed re-encryption graph $\mathcal{DRG}$ which tracks update token queries. $\mathcal{C}_{\mathsf{chal}}$ contains outputs of the challenge oracle $\mathsf{O}_{\mathsf{challenge}}$ as well as outputs of $\mathsf{O}_{\mathsf{ReEnc}}$ when given a challenge as input. $\mathcal{K}_{\mathsf{corrupted}}$ contains the outputs of $\mathsf{O}_{\mathsf{Corrupt}}$ queries. $\mathcal{DRG}$ consists of nodes v_i that represent key pairs, and edges $\overrightarrow{e}_{i,j}$ which are added when $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$ is queried. Using update tokens, the adversary can locally re-encrypt challenge

ciphertexts. Therefore, if a challenge ciphertext is and encryption under pk_i, and there exists a sequence of tokens going from i to j, then both sk_i and sk_j are considered challenge keys. Represented using the graph $\mathcal{DRG}$, if there is a path from v_i to v_j and sk_i is a challenge key, then so is sk_j. At the end of the game, $\mathcal{DRG}$ is used to update the list of challenge keys. Then the trivial winning condition translates $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupted}}$ being empty.

E Adaptive Post-Compromise Security

Here we give the explicit definition for adaptive PCS.

$\mathsf{ad\text{-}PostComp}^{b,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$

$\mathcal{K}_{\mathsf{corrupted}}, \mathcal{K}_{\mathsf{chal}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{DRG} = \emptyset$
$\kappa = 0, \mathsf{called} = \mathsf{false}$
$b' \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{Corrupt}}, \mathsf{O}^{\mathsf{aPC}}_{\mathsf{ReEnc}}, \mathsf{O}^{\mathsf{aPC}}_{\mathsf{challenge}}}(1^\lambda)$
$\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
if $\mathcal{K}_{\mathsf{corrupted}} \cap \mathcal{K}_{\mathsf{chal}} \neq \emptyset$:
  return $\perp$
else return b'

$\mathsf{O}^{\mathsf{aPC}}_{\mathsf{ReEnc}}(C, i, j, [\Delta_{i,j}])$

if $|C_0| \neq |C_1|$ **OR** $\mathsf{called} = \mathsf{true}$: **return** $\perp$
if $(i, C_0), (i, C_1) \notin \mathcal{C}_{\mathsf{honest}}$: **return** $\perp$
if $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$
if not given : $\Delta_{i,j} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C)$
if $(i, C) \in \mathcal{C}_{\mathsf{honest}}$: $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C')$
if $(i, C) \in \mathcal{C}_{\mathsf{chal}}$:
  $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C')$
  $\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_j)$
return C'

$\mathsf{O}^{\mathsf{aPC}}_{\mathsf{challenge}}(C_0, C_1, i, j, \Delta_{i,j})$

if $|C_0| \neq |C_1|$: **return** $\perp$
if $(i, C_0), (i, C_1) \notin \mathcal{C}_{\mathsf{honest}}$: **return** $\perp$
if $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$: **return** $\perp$
$C' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, C_b)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\ (j, C'), \mathcal{C}_{\mathsf{chal}}.\mathsf{add}\ (j, C'), \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\ (\mathsf{sk}_j)$
$\mathsf{called} \leftarrow \mathsf{false}$
return C'

Figure 18: The adaptive post compromise game PostComp. This is stronger than the selective version as the adversary can choose to corrupt keys as a result of challenge queries, subject to the trivial win condition.

Definition 14. *A PRE scheme $\mathcal{PRE}$ is said to have* ϵ-Adaptive Post-Compromise Security (ϵ-adPCS) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left|\Pr\left[\mathsf{ad\text{-}PostComp}^0_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{ad\text{-}PostComp}^1_{\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \epsilon,$$

where $\mathsf{ad\text{-}PostComp}\mathcal{A}^{b,\mathcal{PRE}}$ *is defined in Figure 18. If* ϵ *is negligible as parameterised by the security parameter, then we say the scheme has* Adaptive Post-Compromise Security (adPCS).

F Weak key privacy of our construction

Theorem 7. *The scheme given in Figure 8 has weak key privacy.*

Proof. Once again, the proof replaces the public key and $\mathsf{ReSample}$ outputs associated to identity labels $i = 1, \ldots, \kappa$ by uniform values one by one. This is permissible due to the RLWE assumption. Define the following sequence of games:

- Game_0: The $\mathsf{weakKP}^{b,\mathcal{PRE}}_{\mathcal{A}}(1^\lambda, 1^\kappa)$ game.
- Game'_i for $i = 1, \ldots, \kappa$: The same as Game_{i-1} apart from the fact that pk_i is replaced with a uniform value.
- Game_i for $i = 1, \ldots, \kappa$: The same as Game'_i except that the output of calls to $\mathsf{ReKeyGen}(\cdot, \mathsf{pk}_i)$ are replaced by a uniform value.

The transitions or game hops occur in the order $\mathsf{Game}_0 \to \mathsf{Game}'_1 \to \mathsf{Game}_1 \to \cdots \to \mathsf{Game}'_\kappa \to \mathsf{Game}_\kappa$. We first note that the secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_\kappa$ are never used in any of the games so we ignore these throughout this proof. Game_{i-1} can be seen to be indistinguishable from Game'_i by acknowledging that the difference between these games is that a single RLWE public key is replaced by a uniform value. Next we need to argue that Game'_i is indistinguishable from Game_i. To do so, we consider a sequence of hybrids between Game'_i and Game_i denoted as $\mathsf{Game}'_{i,j}$ where $j = 0 \ldots \lceil \log_2(q)/r \rceil$. Essentially, in $\mathsf{Game}'_{i,j}$, the first j calls of the form $\mathsf{ReSample}(pk_i)$ are replaced by uniform random values. Recall that for both these games, the public key $\mathsf{pk}_i = (a_1, a_2)$ is a uniform random value itself. Therefore, the difference between $\mathsf{Game}'_{i,j}$ and $\mathsf{Game}'_{i,j+1}$ is that the $(j+1)^{th}$ call to $\mathsf{ReSample}$ denoted by (b_1, b_2) either has the form $(a_1 v + e', a_2 v + e'')$ or takes the form of a uniform random value. Considering the pairs (a_1, b_1) and (a_2, b_2), an adversary cannot distinguish between $\mathsf{Game}'_{i,j}$ and $\mathsf{Game}'_{i,j+1}$ with non-negligible advantage according to the RLWE assumption. This holds for $j = 0, \ldots, \lceil \log_2(q)/r \rceil - 1$. Since we have $\mathsf{Game}'_{i,0} = \mathsf{Game}'_i$ and $\mathsf{Game}'_{i,\lceil \log_2(q)/r \rceil} = \mathsf{Game}_i$, we can conclude that Game'_i is indistinguishable from Game_i by using the RLWE assumption multiple times. Iterating through $i = 0$ to κ completes the proof. □